EP 2742 1(3)

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 9/44 | A2

(11) International Publication Number: **WO 00/67117**

(43) International Publication Date: 9 November 2000 (09.11.00)

(21) International Application Number: PCT/US00/11720

(22) International Filing Date: 27 April 2000 (27.04.00)

(30) Priority Data:
09/303,298 30 April 1999 (30.04.99) US

(71) Applicant: PEOPLESOFT, INC. [US/US]; 4460 Hacienda Drive, Pleasanton, CA 94588 (US).

(72) Inventors: LEVY, Martin, J.; 946 San Tomas Aquino Road, Campbell, CA 95008 (US). VIJ, Rajiv; 2285 Bentley Ridge Drive, San Jose, CA 95138 (US).

(74) Agents: SACHS, Robert, R. et al.; Fenwick & West LLP, Two Palo Alto Square, Palo Alto, CA 94306 (US).

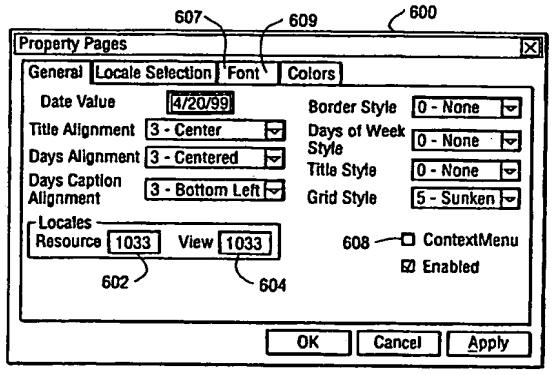
(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

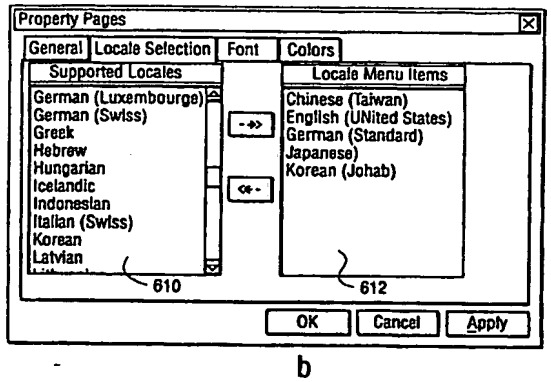
(54) Title: MULTILINGUAL COMPONENTS FOR APPLICATION DEVELOPMENT PROVIDING LOCALE SENSITIVE AND CULTURALLY APPROPRIATE PRESENTATION OF DATA

(57) Abstract

Multilingual components support both development and dynamic runtime selection of locales to automatically update the language and formatting requirements of presented or stored data. Each multilingual component stores information identifying a plurality of different locales that are supported by the multilingual component. During development, the applications developer selects a set of the locales that are to be supported at runtime by the multilingual component, and defines additional user interface and behavioral properties. The developer further specifies an initial one of the locales as the locale which defines the language and presentation formatting requirements for presenting data within the multilingual component. During runtime the developer selected locale is used to present user input (or statically defined) data in the language and format of the locale. If enabled, the user can select a different one of the supported locales, in which case the multilingual component automatically changes the language and formatting of the presented or stored data to that of the user selected locale. The multilingual components include various implementations, including visual components, such as a calendar, a date/time display, a numeric display, an edit display, and a currency display. Non–visual components include a locale component which stores language and formatting parameters for one or more locales, and a text string manipulation control.



a



b

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

MULTILINGUAL COMPONENTS FOR APPLICATION DEVELOPMENT PROVIDING LOCALE SENSITIVE AND CULTURALLY APPROPRIATE PRESENTATION OF DATA

Inventors: Martin Levy and Rajiv Vij

Background

Technical Field

The present invention relates to software components and development environments for developing software applications, and more particularly to software components for use in developing internationalized software applications useful for operation in different national locales having various language and cultural requirements for presentation of information.

Background of Invention

Most large software companies have a need to internationalize their software applications for distribution in many different countries. Many countries may have a specific native language, data formats, and other culturally specific data presentation formats. For example, the language of the user interface (e.g. menus, dialog boxes, message boxes) and underlying data should be a native or understood language for a locale. Further, a locale or country may have specific data presentation conventions. For example, in the United States, commas are used to separate thousands in numbers, and periods are used to identify the decimal place; in most European countries the reverse is true. Likewise, in the United States dates are written typically in MM/DD/YY format, while in Europe dates are typically in DD/MM/YY format. Also each country may have conventions pertaining to general numeric or text data, or to specific types of data such as time, currency, addresses, telephone numbers, and the like. Thus, in order to provide internationalized software for distribution in many different countries, a software vendor must modify a software product to operate in each supported country or locale. To provide a single software product that is truly multilingual and internationally distributable, that is, that can be adapted to operate in different locales, is a considerable challenge.

The traditional practices of internationalization are based on the premise of a global or holistic approach to converting or modifying an application to a particular set of cultural and

language requirements. That is, the software application is viewed as a single entity all of elements of which--menus, keystrokes, dialogues, message boxes, data formats, and the like--must be consistently and uniformly changed. This approach in turn leads to a development program involving making changes to the application's source code in-line. That is, the underlying software code for the entire application must be modified to reflect the language and formatting requirements of a specific country or countries. The modified code is then compiled and distributed in the specific countries. This time consuming process must be repeated for each different locale in which the software is to distributed. As a result, the many different versions of the software must be separately managed and documented, increasing the cost and complexity of distributing the software.

However, new code developed by developers not versed in the art of internationalization will sometimes not be able to operate correctly in a different language environment other than the environment it was developed on, and require considerable reprogramming. Of course, documented standards do help considerably, but unless all aspects of multilingual support are clearly defined, the software has less of a chance of being totally portable across all languages. In all cases, the burden is placed on the developer to figure out exactly what to do in each specific instance of code development. Whether or not the project schedule for a software products allocates enough time for this task is can be a large factor in the amount of consideration paid to the internationalization of the code. Further, even if the resulting product operates correctly, its resulting functionality is now bound to the specific language or formats for which is was modified.

In addition, sometimes it is desirable for the user to be able to change cultural and data presentation features of a software product during runtime. A user fluent in multiple different languages may desire to change the language of the menus and dialogs of an application during operation. This is typically not possible. Instead, the user is at best allowed a limited ability to change only the language and format of data that is input into a program (e.g. text that is typed into a word processing document, or data entered into a spreadsheet), but not the language, formatting of the menus, dialogs, and other software elements of the application itself. This normally requires the software application to be restarted. In other cases, the user can change the overall language and data formatting for all applications, but this requires changing an operating system parameter and rebooting the computer. Again, changing the formatting of an individual component of a software application is conventionally not supported.

While most operating systems and development platforms provide varying levels of international support there is no real adoption of any particular standard or proprietary method for utilizing any of these services. Even to program such support on one particular operating system or environment requires considerable time and effort in order to understand its capabilities and limitations.

While there are many different internationalization APIs available on various operating systems and development platforms, actually utilizing them for multilingual applications is a considerable challenge for software developers. Considerable time must be spent in order to understand how the particular API works and performs as well as understand where the limitations are.

Accordingly, it is desirable to provide a new approach to the internationalization of software products, one that supports the application developer in readily creating software applications that can be easily adapted for specific locales without in-line code changes or interfacing with complex operating system or internationalization API requirements. In addition, it is desirable to provide an approach that allows a user to dynamically modify the cultural presentation of discrete components of a software application dynamically, without restarting the application, or rebooting the computer.

Summary of Invention

The present invention overcomes the limitations of conventional approaches to the internationalization of software, is responsive to the cultural needs of software systems and software users, and eliminates the necessity of rewriting software components for every different cultural environment. The present invention does this by providing intelligent multilingual components that can dynamically modify their language and their data formatting parameters in response to a variety of different cultural environments, or more specifically in response to change in a currently selected locale. The selection of locale and the change in language and formatting parameters can be made both at development time initially by the applications developer, and during runtime operation by the user. The multilingual components are independent from the software application that uses them, and may be reused in different applications.

The cultural requirements for software can be broken down into small discrete multilingual components that contain embedded knowledge and behaviors about the language and presentation formats of a variety of different cultural domains or locales. These

components are then used by the applications developer as building blocks for creating completely dynamic multilingual software applications. In addition, consistent culturally correct results can be achieved by centrally coupling certain processing events to be handled by the multilingual components to available system or application level data descriptive of the current locale.

By condensing the dynamic multilingual support into intelligent multilingual components, multilingual support and code reuse is realized. The provision of multilingual components that concentrate the cultural knowledge and understanding into discrete and manageable units, eliminates hours of source code modifications and maintenance overhead. The burden of internationalization programming is then removed from the application and systems developer.

In the present invention, each multilingual component can be developed and maintained individually. A multilingual component contains the required knowledge to perform its function in any of a target environment's supported languages.

Each multilingual component provides services for formatting, interpreting and presenting locale sensitive data within a software application. A multilingual component is sensitive and responsive to the data identifying the locale in which it is operating, and thus dynamically modifies its language or other cultural parameters in response to this locale data.

The multilingual components provide programming interfaces with callable methods and modifiable properties that condition, control and alter the component's behavior. The multilingual components utilize a host environment that is able to detect the presence of the multilingual components and then support two-way communication to such components via events and messages. The host environment conveys certain basic ambient conditions that the multilingual components abide by such as font styles, colors, border style, appearance (3d or flat), background color, foreground color, and font.

Each multilingual component is responsible for property persistence per instance of utilization. This allows the same type of multilingual component to be used in multiple scenarios in the same application while at the time retaining their individual instance configuration. For example, a currency multilingual component can have multiple appearances in an application in different windows or message boxes, and each appearance can be defined as having different language and culturally specific presentation formats. Thus, one instance of a currency multilingual component may be formatted with the English language style comma and period separation of numbers and "$" for currency, and present its data in

that format, while another currency multilingual component in the same application may be formatted for a specific European currency symbol and number format. Each instance would maintain its formatting features over time until changed by the user or underlying environment information.

Preferably, a locale multilingual component is provided that supports the definitions of the language and formatting requirements for each locale; other multilingual components refer to the locale multilingual components to obtain the necessary formatting and language information. A locale multilingual component obtains the underlying data from the operating system; this information is overridable by the developer if desired.

The host environment is able to programmatically query each multilingual component in an application to determine a particular component's interfaces and property set. The host environment uses this capability to support dynamic manipulation of the entity at runtime through application specific means (scripting, dialogs, menus, etc.).

Ultimately, the software application is able to notify the multilingual component of changes in the application's active language state in order to cause the multilingual component to modify its presentation and result set in accordance with the cultural conventions of the specific language request. Multilingual support is based around a locale model that is used to identify particular languages and drive the multilingual components.

Multilingual components can either use the host platform operating system's national language support primitives or any other proprietary API that provides the required international support.

The utilization of multilingual components for software application development shifts the burden of significant internationalization development, maintenance and support by shifting the locale specific responsibility of multilingual support for data formatting, interpretation and presentation to the multilingual components themselves. The multilingual components are maintained independently of any software application and provide a consistent trusted set of interfaces, methods and properties that can relied upon to manage all related multilingual operations.

Multilingual components provide several layers of interfaces that share many common multilingual services. Some the services and functionality that these multilingual components may provide include:

- Automatic font determination based upon the active or specified locale and the available system resources.

- Automatic data formatting parameters for the active or specified locale.
- Automatic culturally sensitive parsing and data interpretation for the active or specified locale.
- Automatic localized message retrieval based upon of the active or specified locale.

The behavior of a multilingual component may be changed according to rules that can be:

- Configured for each multilingual component before being used (configuration properties).
- Programmed or scripted while the multilingual component is active at runtime.
- Set by the user while it is executing the multilingual component through the use of a built-in, on-demand, localized, language sensitive user interface.
- Determined from the user's application environment while the multilingual component is executing.

Multilingual components are small and discrete and can be used as building blocks to construct more complex multilingual components and applications. A multilingual component can work in conjunction with other multilingual components.

The multilingual components may be used in a visual and non-visual mode. This means that they can be used to interact with a software application's end user for data presentation and data entry as well as be used to perform the associated multilingual support services for any requested locale on demand.

As long as the host applications can support the particular technology and programming language selected to implement the entities, the multilingual components can be used in multiple software applications.

During application development, the multilingual components are used as follows: The developer generally creates a container window of a client application and hosts one or more of the multilingual components in this container window by instantiating the multilingual component and attaching it to the container window. Thereafter the developer can manipulate any properties or methods that the multilingual component exposes. The developer may use the multilingual component's interface in a non-visual way by just instantiating the component and calling methods on it. The multilingual components can either be made available in a commercially or proprietary available rapid application development environment that supports the language and communications mechanism in which the multilingual components have been implemented. A multilingual component can usually be added to a container

window via a menu user interface. If the application developer wants to use the services of an multilingual component that does not have a visual presentation, a reference to the multilingual component can be set in the application and its services can be called by querying for an interface and calling methods.

In a preferred embodiment, the multilingual components can be utilized in two major ways:

1. Each multilingual component provides a simple interface of methods and properties (API) that can be used to set and get data on the component in a culturally correct manner for the particular language that the components has determined to be active and in effect for itself according to any of a number of property rules.
2. In addition to #1, the multilingual component is capable of visually receiving and presenting data interactively through the normal means of user data entry within a software application.

Essentially, the entity is able to operate in both a visual and non-visual manner.

Examples of such multilingual components include, but are not limited to the following:

Edit

Provides a multilingual text data entry and presentation environment. Supports Asian input method editors and modes, bi-directional text processing, font changes, etc.

Numeric

Provides a multilingual numeric data entry and presentation environment. Supports Asian input method editors and modes, correct locale numeric formatting, etc.

Currency

Provides a multilingual monetary data entry and presentation environment.

Time/Date

Provides a multilingual time & date data entry and presentation environment.

Calendar

Provides a multilingual calendar for date data entry.

The present invention provides numerous advantages over conventional approaches to internationalization. Developing and maintaining individual software components that can be tested in isolation from any of the hosting applications is an efficient mode of software development. Shifting complex multilingual knowledge and requirements, normally required of a software application, to the internals of the multilingual component alleviates the burden

from software developers and allows developers to focus on their primary domain of expertise, creating the application itself, rather than attempting to internationalize the application.

Description of the Drawings

Figs. 1a-1d are illustrations of a calendar multilingual component.

Figs. 2a-2b are illustrations of a date/time multilingual component.

Fig. 3a-3b are illustrations of a currency multilingual component.

Figs. 4a-4b are illustrations of an edit multilingual component.

Fig. 5a is another illustration of an edit multilingual component.

Fig. 5b is an illustration of the property sheet of an edit multilingual component.

Figs. 6a-6b are illustrations of the property sheets of a calendar multilingual component.

Figs. 7a-7b are illustrations of a development environment supporting the multilingual components of the present invention.

Fig. 8 is an illustration of the modules for one implementation of the calendar multilingual component.

Fig. 9 is a flowchart of the input reaction unit of the calendar multilingual component.

Fig. 10 is a flowchart of the calendar rendering engine of the calendar multilingual component.

Fig. 11 is a flowchart of the selector control of the calendar multilingual component.

Detailed Description of the Preferred Embodiments

The present invention supports both visual and non-visual multilingual components, the former type having a user interface for presentation of data to, and receipt of data from, the user; the latter which operate without direct presentation to the user, but which store and operate on user or application provided data, or which provide services to other multilingual components.

Referring generally now to Figs. 1a through 3b, there are shown various illustrations of multilingual components having visual presentations. Beginning with Fig. 1a, there is shown a calendar multilingual component 100. As with all multilingual components of the present invention, the calendar multilingual component 100 is a reusable component, allowing it to be used in different applications by an applications developer as needed, and altered by the developer for the particular application. The calendar multilingual component 100 provides the functionality of multi-year calendar, which can be used in a software program to enable the user to quickly select dates from current, past, or future months, and provide the selected date information to other functional aspects of the software program. The calendar multilingual component 100 includes a month name area 104, a day name area 106, along with standard layout of dates for corresponding months.

In addition, the calendar multilingual component 100 includes a context menu 102 which lists the various locales, here identified by a combination of language, (e.g. "English") and region (e.g. "United States") or type (e.g. "Standard") that are supported by the calendar 100. The available languages are chosen by the applications developer from a total list of languages supported by the calendar multilingual component. When the user selects a particular locale from the context menu 102, the month name area 104 and day name area 106 are updated to reflect the name of the shown month and the names of the days in the language associated with the selected locale. This selection of locale and language for the calendar multilingual component 100 occurs independently of, and may be different from, the language designation (if any) of any other multilingual component that the software program (client application) supports.

Fig. 1b now shows the same calendar multilingual component 100 after the user has selected Japanese as the desired language. The month name area 104 and the day name area 106 have each automatically updated themselves to reflect their respective data in the Japanese language. This includes selection of both appropriate font types and sizes, and selection of

correct day and month names (since mere change of font from English to Japanese would not effect the correct change in names).

Figs. 1c and 1d illustrate further changes in the same calendar multilingual component 100, with Fig. 1c illustrating the calendar multilingual component in the French language and showing the context menu designation, and Fig. 1d illustrating the change in the calendar multilingual component to Hebrew language. Again, in each case the name of the month and days has been changed, in addition to proper selection of the appropriate font for the language.

Figs. 2a and 2b illustrate a date/time multilingual component 200 operating in conjunction with a calendar multilingual component. Fig. 2a illustrates the component 200 with English date/time formatting, i.e. MM/DD/YY. Also shown is the context menu 202. Note that here, there are a variety of locales that share the English language, but which differ by region (e.g. United States, United Kingdom, etc.). Fig. 2b shows another date/time component 200, but after the user has selected, via the context menu, German as the locale specific language.

Figs. 3a and 3b illustrate a currency multilingual component 300 in its various presentations of different currency data responsive to locale selection by a user. In Fig. 3a there is shown the currency multilingual component 300 with its context menu 302 showing the available locales that are supported. Here, the user has selected "German (Standard)"as the appropriate locale. The currency multilingual component 300 includes the data field 304 in which the currency data 304 is shown with the appropriate locale specific numeric formatting, and with the appropriate currency symbol 306. Fig. 3b illustrates that selection of another language for a different locale, e.g. Japanese, results in the currency multilingual component 300 automatically reformatting the currency data 304 and selecting the appropriate currency symbol 306 (Yen) for the locale associated with the selected language. In one embodiment, the component provides a conversion between currencies using a predetermined conversion ratio, e.g. 1:1. In another embodiment, the conversion ratio is dynamic and determined at the time the user invokes the change of currency, such as looking up the conversion ratio in a database or table which stores frequently updated conversion ratios.

Fig. 4a illustrates an edit multilingual component 400. From the user's perspective, the edit control 400 is simply a text box in which text may be entered. However, unlike conventional text boxes it supports a context menu 402 which allows the formatting and presentation to be dynamically altered for different locales. Thus, Fig. 4b illustrates the same text box, after the locale has been changed from English to German.

Having given these various examples of the multilingual components of the present invention, the following now describes an implementation of the multilingual components in further detail.

I. Overview of Implementation

The multilingual components of the present invention operate in a conventional computer system, including a processor, memory, disk storage, network connection, and operating system environment. The operating system preferably supports a windowing environment. A suitable operating system includes Microsoft Corp.'s Windows95, WindowsNT and the like. In a Microsoft Windows95 implementation, the multilingual components of the present invention are embodied as ActiveX controls and objects; however, the present invention may be implemented as other types of entities, for example as classes in object oriented environments, such as Java, C++, SmallTalk, Eiffel, and the like. In other embodiments, the multilingual components may operate in a network computer running a browser environment for connection to a server providing the multilingual components. ActiveX implementations operate in Microsoft's Internet Explorer, or other browsers supporting ActiveX plugins.

In accordance with the present invention, the multilingual components are reusable components, such as ActiveX controls, that provide a collectively common set of internationalization services via their interfaces and a high level of flexibility in controlling the component's behavior, dynamically, via a set of configurable properties and methods. These multilingual components can be used both by the developers and end users. The multilingual components can be used both as container-embedded controls or as foundation controls to exploit their services in non-visual applications. For the purposes of this specification the terms component and control are used interchangeably.

In general the multilingual components are locale sensitive and are driven by a locale property. A multilingual component can simultaneously represent data in any supported locale just by modifying its locale property. This modification can take place programmatically or a user can interact with the multilingual component at runtime and decide to change the locale and language in which the user wants to view the data. For instance a user viewing a numeric value on a numeric multilingual component, formatted for a German locale, can instantly choose to view it formatted for the U.S. locale by altering the locale via the context menu. The

output in the German locale will display the thousands separator as a period, ".", with the comma "," as the decimal point, and changing to U.S. locale will automatically display the comma as the thousands separator, and the period as the decimal point.

For the purposes of this specification, the following definitions are applicable:

Container: A client application that is capable of embedding an multilingual component.

Data Locale: The locale attribute associated with a piece of data. For example, an item of currency data stored in a database or application may have a German locale associated with it, representing a German currency, but be presented in a currency multilingual component which has a different locale, such as Japanese.

Locale ID: A value defined by the operating system that uniquely identifies a locale. The locale determines the language/country/culture conventions such as number/currency/date/time formats.

Locale Sensitive Properties: Multilingual component properties that are changed by the component when the underlying Locale ID is altered. The locale specific properties include the language and formatting parameters of the component.

View Locale: The locale to use for formatting and presenting the data to the user. In contrast to the Data Locale which is associated with data, the View Locale is used to present the data in an alternative locale, instead of that of the Data Locale. In the example of Figs. 3a and 3b above regarding the Data Locale for a currency item, the View Locale is Japanese.

Resource Locale: The Resource locale dictates the language/country/culture conventions in which the error messages and string resources of the application will be displayed. For example a currency multilingual component which displays U.S. currency data, may display an error message in Korean, on invalid input entry, if the resource locale is set to Korean. The string resources are those that are used to display text and other data of the multilingual component itself, such as its menus, title bars, dialogues, and other display areas. A multilingual component could operate in multiple languages, but it is useful to provide a neutral language to display string and menu resources.

Significant Locale: A significant locale is conceptual construct describing which locale is most significant for a particular component. The significant locale may either be the Data Locale property or the View Locale property depending upon the component. The selection of one of these locales as the significant locale drives the locale sensitive properties to alter the presentation of the component.

II. Basic Multilingual Component Characteristics

In a preferred embodiment, all multilingual components comprise common features including stock properties and events.

Containment of Locale Multilingual Component

In a preferred embodiment, each instance of an multilingual component is associated by containment with one or more locale multilingual components. Each locale multilingual component encapsulates the locale specific properties and requirements of a particular locale. A multilingual component obtains this information from the locale multilingual component interface.

Dual Interfaced

For an ActiveX embodiment of the multilingual components, it is preferred that all components are dual-interfaced, i.e. they can be driven either through an IDispatch interface type invocation, which allows late binding of object methods, or can be manipulated directly through a virtual function table. This allows the component to be used efficiently from high level languages like C++ which access the virtual function table directly, rather than accessing properties and methods from the Dispatch interface, which can be slow.

Error Handling

In a Windows embodiment, error messages are generated via the OLE Dispatch Exception mechanism.

Error messages are dispatched in the language determined by the Resource Locale property. Error messages are distributed in a specific DLL for a language. The error messages are loaded from this DLL when the application is executed. If the DLL does not exist, the message as a last resort is displayed in the default resource language.

Locale Driven

The locale sensitive properties of a multilingual component are driven by a multilingual component's view or data locale. Modifying one of these locale properties result in an appropriate alteration of the locale sensitive properties.

Control Service Access

Some controls support interfaces for user interaction support. Where the multilingual component are implemented as ActiveX controls, then all component services should be accessible via an IUnknown Interface. Since the controls also support automation the methods and properties can be accessed via the IDispatch interface.

Control License

In one preferred embodiment, all multilingual components implement a licensing interface, so that their use is legalized. A client running an multilingual component has to obtain appropriate licenses for using the multilingual component .

Internet Ready

All multilingual components are Internet ready and can be downloaded over the Internet. In an ActiveX implementation, the controls implement the IObjectSafety interface for determining whether the control is safe for scripting or initializing.

MFC & ATL Based

In one embodiment, the multilingual components are written in C++ and use the Microsoft Foundation Class DLL's. The non-visual objects may be written using Microsoft's ATL SDK.

III. Visual Multilingual Component Stock Properties

Table 1 lists the preferred stock properties that are implemented by the visual multilingual components..

Table 1: Stock Visual Component Properties

Property Name	Type	Description
Alignment	Integer	0 - Left justify text 1 - Right justify text 2 - Center text
Appearance	Integer	0 - Flat. Paints control without visual effects. 1 - (Default) 3D. Paints control with three-dimensional effects.
BackColor	OLE_COLOR	Sets the Background color of the control.
BorderStyle	Integer	0 - The control has no visible border line. 1 - The control has a single-line border (default).
Enabled	Boolean	Enable/Disable the control.
Font	IFontDispatch	Set's the Control's Font.
ForeColor	OLE_COLOR	Sets the foreground color of the control.
Text	String	Returns/Sets text contained in the control object.
ShowToolTip	Boolean	Enable or Disable the Tool Tips for the control. (Default - True)
AutoKBLayout	Boolean	Determines whether an individual

		control on a form changes the Keyboard Layout to that based on the Locale property. This allows each multilingual component to specify, if desired, its own keyboard layout. (Default - True)
ContextMenu	Boolean	Enable or Disable the default menu associated with the control. Is dependent on the control type. The menu allows the user to change the locale information for the control. (Default - True)
ViewLocale	Integer	LocalID of the locale to use for language/country/culture formatting and presentation requirements for presenting data to the user.
ResourceLocale	Integer	LocalID of the locale to use for the language/country/culture conventions in which the error messages and string resources of the application will be displayed.
Value	Dependent on type of component.	Default data to be included in component, such as default text for an edit component, or default number for a currency component.

As those of skill in the art will appreciate that one or more of these properties may be optionally eliminated from an multilingual component while still providing the essential functionality of the present invention. Likewise other default properties may be added.

Fig. 5b illustrates a sample property sheet 512 for accessing the properties of a multilingual component directly by the applications developer. Here, the property sheet 512 is for an instance of an edit multilingual component is shown. Fields 514 here allow the developer to define the locale IDs of the Resource and View locales. These locale IDs reference specific locale multilingual components which embed the locale specific information for their respective locales. Other fields allow the developer to specify, for example, whether the edit is a multi-line or not (516) or read-only (518).

Fig. 5a illustrates an example of an edit component itselfr. Via the miscellaneous setting checkboxes 502, one can enable or disable various ones of the above mentioned properties. Default text is set in text box 506. The user can select the locale and language, using context menu 508. Updates to these setting are effected by update button 510.

Figs. 6a and 6b illustrate the property sheet 600 for an instance of a calendar multilingual component. In Fig. 6a, the basic organization of the calendar is defined by the applications developer, including alignment, styles, grid type, and so forth. The context menu for allowing user alteration of the locale can be enabled at checkbox 608. In addition, the developer specifies the locales by LocaleID for the ResourceLocale (602) and the ViewLocale (604). These locales will be used, as described above, when the calendar multilingual component attempts to determine which LocaleID to use.

In Fig. 6b, the developer selects from a list 610 of supported locales those that will be available to the end user, and places those selected locales in the locale menu items list 612. These selected locales will appear on the context menu if enabled. The non-selected locales are not available to the user. This allows the developer to specifically control which locales will be supported at the user level, thereby limiting, if desired, the user's ability to dynamically change the locale and presentation of the component. The applications developer is freed from having to define the locale specific information, as this is encapsulated in the multilingual component as provided to the developer.

As indicated in Fig. 6a, the property pages include a font page 607 (itself not shown) for selecting a default font for the presentation of the component; a font selection list of available system fonts is provided with standard mechanisms for selection. A color page 609 (not shown) provides standard mechanisms (e.g. color palette or wheel) for selecting a default color for the component.

For other types of multilingual components, the property specification, including identification of resource and view locales, and selection of available locales, fonts, and colors, are essentially the same as illustrated in Figs. 6a and 6b.

IV. Basic User Interaction

Each visual multilingual component supports additional useful behavior in response to various user actions. First, each multilingual component preferably supports a tool tip. If the cursor is pointed on the multilingual component, a tool tip appears, indicating the currently selected language for the chosen multilingual component. The appearance of the tool pip can be enabled or disabled via a ToolTip property by the developer or application user.

As shown above, it is preferable to allow the user to change the ViewLocale of a multilingual component by using the context menu to displays the available input locales for

the chosen control. However, this feature need not be always made available to the user, and can be enabled via an ContextMenu property.

When the user changes focus from one multilingual component to the other, the keyboard input layout changes, to reflect the input locale, determined by it's significant locale. This feature can be enabled via the AutoKBLayout property.

Display formatting, if any, is done only after a multilingual component loses focus. Before the user switches focus out from the current multilingual component, a LostFocus event is sent to the container, which then checks if the input is valid and determines whether can switch focus out of the multilingual component. If so, the focus is released, and the container switches focus to the appropriate object.

V. Visual Control Stock Event

The following is the preferred common stock event implemented by the multilingual components that have a visual presentation:

ViewLocaleChange

This event is generated when the multilingual component's View locale (language) is changed via the context menu. During the processing of the ViewLocaleChange, the container may set a new font on the control or perform other related tasks.

VI. Component Containers

The visual multilingual components are embedded in a container, with the container responsible for controlling the component. A container may be any window or visual object of a client application, such as a Visual Basic application. A user may interact with the multilingual component by inputting information or accessing a multilingual component's context menu to change the view locale or other properties.

In an ActiveX implementation, the multilingual components can be hosted in any container that supports the OCX96 specification, as specified by Microsoft Corp.

VII. Visual Multilingual Component Embedding & Initialization Sequence

A multilingual component can be embedded in the container application, for example, by dragging an icon for the multilingual component from a Tool Box over to the container. Each container includes a LocaleID which defines the locale which is used to define the initial local specific properties of its components. Fig. 7a illustrates an exemplary development environment 700, such as Microsoft Visual Basic. Here, the developer defines various

containers, such as form 702, to contain various components, including the multilingual components of the present invention. The multilingual components are included, along with pre-existing components, in a palette 704, or other user interface mechanism. Fig. 7b illustrates the same form 702 after the developer has placed a calendar multilingual component 706 on the form.

When a multilingual component is placed in a container, the multilingual component initializes itself by retrieving the ambient locale Identifier (LocaleID) from the container and setting its ViewLocale with this value. If the container does not support the LocaleID property then the control initializes itself with the user's default locale ID, which is managed by the operating system. If the user's default locale cannot be retrieved, the operating system's locale (regional settings) is used to set both both the ViewLocale and the ResourceLocale (and the DataLocale, if present).

In cases where the data that is used in a multilingual component is retrieved from the application itself, a database, or other predefined source (as opposed to being merely user entered), then the data may have a Data Locale associated with it. In this case, the ViewLocale will remain as set, but the Data Locale is selectively used to format the data. For example, in one embodiment, the currency multilingual component operates on this principle. For example, a client application may store currency data such as ¥100 (100 Yen) in a database as the value 100 and data locale code uniquely associated with Japan (e.g. JPN). When the currency multilingual component is configured, it adopts as its data locale JPN and a distinct view locale, such as USA (U.S.A). During runtime, the component will display the 100 value formatted for the U.S. locale, though displaying the Yen symbol. The ViewLocale can be changed (e.g. to German, for different number separators) while retaining the DataLocale.

Once the ViewLocale is established the multilingual component sets its locale sensitive properties accordingly.

VIII. Container/Client in User Mode (Application running)

When the container application is running it can now set the multilingual component's locale properties. The initial locale sensitive properties of the control are set based according to the ViewLocale ID. The container is responsible for mapping a database field data, or some other data entity to the multilingual component's Value property, if applicable.

Setting one of the multilingual component's locale ID properties transmits a ViewLocaleChange notification to the container. The container event handler code may

decide at this point to change the font of the multilingual component so that the presentation of data is visible in the appropriate font.

IX. Container Responsibilities

A container which stores an multilingual component drives the contained multimedia components based on the locale associated with any data retrieved from a database or other repository used by the container. This means that the container is responsible for maintaining the locale associated with the data and storing it in the database or other persistent data repository. The container is responsible for providing the component with a handle to a property bag or stream for persistently storing its properties.

The container sets the value of the multilingual component by setting the Value property and also setting the ViewLocale property. The container thus is responsible for mapping a database field data to the multilingual component's Value property. It also retrieves the data from the multilingual component as necessary and calls procedures to store it in the appropriate data repository.

A container preferably responds to a multilingual component's ViewLocaleChange event. The application should be able to let a designer, at design time, create a Locale Default Properties Table. This table preferably contains predefined settings that are defaults for the locale, like the font a particular locale should use as a default, among other properties like the style, size, and the like. The container should also have access to such a table. When processing the ViewLocaleChange event the container can decide to set the font to the default that has been allocated at design time.

All strings passed as parameters to multilingual components are preferably converted to a wide (Unicode) format in embodiment where this is an OLE requirement. All strings received by the container, for example, inside an event parameter, are wide format.

For the Date/Time multilingual component, along with the date/time, the container preferably stores a TimeZoneId that is associated with it. The container must be able to bind both the Value & TimeZone properties to a data source.

The container should handle exceptions and events as needed. The container needs to handle any exceptions generated by the multilingual component. In a Windows embodiment, a multilingual component generates errors via the OLE exception mechanism. The OLE exception constitutes an Error Code and an Error String. The Error String is in the language as specified by the container's locale ID. However the Error String may be empty, in case the

multilingual component's resources do not contain an error string for the container language. In any case the container must process the exception in an appropriate manner, which may be as simple as displaying the error string in a message box.

Finally, the container is responsible for determining whether it wants to handle events generated by the multilingual component for the purpose of event handling.

X. Component & Container Boundaries

The container or the multilingual component must not violate each other's boundaries, and the communication between the two should confirm to the application communication standards for the implementation environment, e.g. COM standards for ActiveX controls.

The component and control communication is preferably by events and methods and properties. The following are preferred features of component-container communication:

- The multilingual component should not have to maintain information about where and how its data is stored. The container should have to maintain information about how to get data and set this data in the multilingual component. Likewise the container should also be liable of storing this data if needed.
- The currency control, for instance, need not have access to externally provided currency tables for conversion purposes. If the user changes the ViewLocale, an event gets generated, and the container should manage any conversions, if needed.

XI. Component Persistence

An instance of a multilingual component in a form is persistent, and stores its LocaleID, along with locale sensitive properties and stock properties. When the application is restarted, the multilingual component restores its properties to the state before the application was closed. That is to say, all multilingual components on a form will save and restore their individual properties appropriately.

XII. Multilingual Component Implementation

Having described the implementation features of the multilingual components generally, this section now describes, in further detail, the behavior, properties, methods, events related to the Currency, DateTime, Edit, Locale and Numeric, controls. These details are for one particular implementation as ActiveX controls. Those of skill in the art appreciate that the names, formats, method interfaces and other programming details may be easily changed while preserving the essential features and principles of the present invention.

A) Calendar Multilingual Component

The calendar multilingual component provides a user interface for entering dates in a client application, and propagating such date information to other uses of the client. The calendar multilingual component can be used as a standalone entity or in combination with the DateTime multilingual component.

1) Component Behavior

On instantiation the calendar multilingual component displays the days of the current month in a grid, with the current day highlighted. Initially the ViewLocale of the control defaults to the system default locale. The calendar multilingual component also displays the current month name and year, along with the day names in the language of the locale as implied by the ViewLocale property.

The user can select any day in the month by positioning the mouse on the day and clicking the left mouse button. The user can also move to the next or previous month via the previous/next buttons. Alternatively the user can point the mouse on a bar and bring up a popup context menu which display the previous 3 month and the next three months (in the native language) as items, that can be chosen from the menu, which allows for a fast and efficient navigation into the months/years.

Table 2: Class Name: MC.Calendar Class Properties

Name	Type	Description
BackColor	OLE_COLOR	Sets the background color of the control.
BorderStyle	Integer	0 - The control has no visible border line. 1 - The control has a single-line border (default).
ContextMenu	Boolean	Enable or Disable the default menu associated with the control. Is dependent on the control type. (Default - True)
Day	Integer	Day as a numeral.
Enabled	Boolean	Enable/Disable the control
Font	IFontDispatch	Sets the control's font.
ForeColor	OLE_COLOR	Sets the foreground color of the control.
GridStyle	Integer	0 – None 1 – Flat 2 – Bumped

		3 – Etched 4 – Raised 5 – Sunken
Month	Integer	Month as a numeral.
ResourceLocale	VARIANT	The locale to use for displaying errors and menu resources. This value could be one of the following: · A locale identifier. · A concatenation of language and country abbreviation.
Title	Integer	0 – None 1 – Left Justify 2 – Right Justify 3 – Center
TitleColor	OLE_COLOR	Sets the title color.
TitleFont	IFontDispatch	Sets the control's title font.
Value	OLE DATE	The actual date value represented by an 8 byte OLE Date.
ViewLocale	VARIANT	The ViewLocale that the control represents. This value could be one of the following: · A locale identifier. · A concatenation of language and country abbreviation. · Another locale object.
Year	Integer	Year as a numeral.

Methods: None

Events

ViewLocaleChange

This event is generated when a view locale change has been requested by the user via the context sensitive menu.

DateTimeChange

Param: DATE NewDateParam: DATE NewDate

This event is generated when the user has modified the date via the user interface

PageBrowse

This event is generated when a user selects the forward or back buttons to change the months while browsing months.

Referring now to Fig. 8, there is shown in further detail an exemplary implementation of a calendar multilingual component. The component is implemented as a collection of series of high level modules. These modules are:

The Input Reaction Unit 808 (IRU): The IRU 808 is responsible for handling all end user input via the mouse or the keyboard. For instance a user might right click the mouse button on the component, which the IRU 808 will process to display a context menu containing languages that can be selected by the user. Fig. 9 illustrates the control flow of the IRU 808 in response to a user input.

The Calendar Rendering Engine (CRE) 804: The CRU 804 is the portion of the component that takes care of the display issues and dynamically changing the view to display the calendar in various languages, that are supported by the operating system. The CRE 804 is further responsible for generating the grid layout for the dates. The CRE 804 exposes properties and methods to control the visual aspects of the calendar in the correct underlying language. The CRE 804 draws the elements of the calendar multilingual component in the correct language and cultural conventions. Fig. 10 illustrates the normal drawing sequence, including calculating 1002 settings for various elements, such as grid and border location, drawing 1004 calendar borders, drawing 1006 the month name in the correct language as determined by the locale, drawing 1008 days of the week in the correct language, and drawing 1010 the grid elements themselves.

The Selector Control Module (SCM) 802: When the user clicks the mouse in the title area 104, the IRU 808 sends the request to the SCM 802 to display a selector control containing months for the past and future months. From there on the SCM 802 communicates with the CRE 804 and the Calendar Events Generator 806 to process user requests. Fig. 11 illustrates the general behavior of the selector control. The selector control pops up 1102 when selected. When the control receives 1104 a user selection of month, the Calendar Events Generator 806 is notified 1106. The CRE 804 renders 1108 (see Fig. 10) the calendar for the selected month, applying the appropriate locale format requirements, such as font, language, and so forth.

In a preferred embodiment, the selector control is a Windows class, which behaves like a menu control, and allows scrolling of items in both directions. The scroll speed is determined by whether the mouse pointer is in the warm or hot scroll regions. The selector control is composed of the selector window, which displays the selector items, and a selector list, which holds the items to be displayed. The selector window displays a subset of the items

held in the selector list. Items can be appended or inserted via sending the SLRM_APPEND and SLRM_INSERT messages to the control. The message SLRM_SETVISIBLEITEMCOUNT determines the number of items that will be displayed in the selector window from the selector list. Further on, the message SLRM_SETTOPINDEX can be send to the control to determine the index in the selector list from which the selector control must start displaying the items.

The Calendar Events Generator (CEG) 806: This module is responsible for firing any events to the container of the calendar multilingual component. These events include DateChange, PageBrowse and the ViewLocaleChange events.

The other types of multilingual component have similar implementations, but typically simpler, such as without the selector control, and a simpler input reaction unit, specific to the types of inputs that the component receives.

B) Currency Multilingual Component

A currency multilingual component exposes provisions to manage attributes that determine the formatting of a currency. It is a visual control that allows a user to enter a currency value.

1) Component Behavior

The currency multilingual component provides validation by letting the user input only numbers, currency sign and the decimal separator string. The user can decide which locale's formatting should be used to display the currency value. This is achieved by the user using the context menu of installed locales to choose from (the installed locales were illustrated, for example, in Fig. 6b). The control display is formatted according to the ViewLocale when the control looses focus.

Changing the ViewLocale resets the underlying currency formatting properties. By default the DataLocale and the ViewLocale properties represent the same Locale. However if the ViewLocale is changed by the user, the formatted currency output is displayed with the ViewLocale property's attributes and the DataLocale property's symbol.

If the user alters the ViewLocale, a ViewLocaleChange event is generated to the container. The container can do any currency conversions at this point and set the new value via the Value property.

The underlying value of the currency is set under two conditions:

1. Directly by setting the Value property; or
2. Implicitly after the user has entered data in the component, and it has been successfully parsed.

2) Class Definition —

The following provides a sample class definition for the currency multilingual component.

Table 3: Class Name: MC.Currency Class Properties

Property Name	Type	Description
BeepOnError	BOOL	Beep on validation errors. (Default - False)
DecimalSeparator	String	The decimal separator string used to display the output.
FractionalDigits	Integer	The number of fractional digits to display for the view locale.
GroupingSize	Integer	The size for each group of monetary digits to the left of the decimal separator.
IntlSymbol	BOOL	If True displays the international symbol characters specified by the ISO 4217. (Default - False)
LeadingZeros	Integer	Specifies the leading zeros in decimal fields. Possible values: 0 - No Leading Zeros 1 - Leading Zeros
DataLocale	VARIANT	The Locale that the control represents. This value could be one of the following: · Locale Identifier. · A combination of Language and Country abbreviation. · Another Locale Object.
LocaleNameMenu	VARIANT	Safe Array holding LocaleID's that will included in the context menu.
NegativeStyle	Integer	One of the 16 negative currency modes to use to display negative monetary values. This value must be between 0 - 15.
PositiveStyle	Integer	One of the 4 positive currency modes to use to display positive monetary values. This value must be between 0 - 3.
ResourceLocale	VARIANT	The locale to use for displaying errors

		and menu resources. This value could be one of the following: · A locale identifier. · A concatenation of language and country abbreviation.
Symbol	String	The Currency symbol represented by the current locale.
ThousandSeparator	String	The thousand separator string used to display the output
UseNativeDigits	Boolean	Indicates whether to display the output using the View Locale's native digits. (Default - False)
Value	double	The actual monetary value represented by the control. (Default - 0)
ViewLocale	VARIANT	The ViewLocale that the control represents. This value could be one of the following: · A locale identifier. · A concatenation of language and country abbreviation. · Another locale object.

Methods: None

Events

ViewLocaleChange

This event is generated when a view locale change has been requested by the user.

C) DateTime Multilingual Component

The DateTime multilingual component is used for input and formatting of date and time strings. The component allows users to format date & time via date/time specifiers. On creation the control value defaults to the current date/time, and the system's default locale, and displays the current date in the formatting convention of the locale. The control can represent either a date or a time at any given instance, or both.

1) Component Behavior

The user inputs the date based on the convention of the locale identified by the ViewLocale property. Date can be input in the form n1/n2/n3, where n1, n2 and n3 are numeric components of the date. Depending upon the locale, n1 could either represent a month or a day, etc. A Date/Time could be entered in the format n1/n2/n3 n4:n5, and so on. The date separator may or may not be "/", and is dependent upon the locale convention, as

identified by the ViewLocale. The time portion of the component encapsulates a time, which can be modified via the Hour/Minute/Second properties.

The DateTime component can be set to display a dropdown calendar, for ease of choosing dates. This calendar is based on the calendar multilingual component.

(a) Input Validation

The Date/Time input is parsed and validated when the control looses focus, and if validated successfully is formatted and displayed according to the Format property. If the validation of input date fails, the date defaults to the value that existed upon the control gaining the input focus.

(b) Output Formatting

The date/time can be customized via the Format property. If the Format property has not been set, changing the ViewLocale (from the context sensitive menu) will modify the Format property, and display the date/time according to that locale. The Format will be set specific to that locale.

Table 4: Class Name: MC.DateTime Class Properties

Name	Type	Description
Appearance	Integer	0 - Flat. Paints controls without visual effects. 1 - (Default) 3D. Paints controls with three-dimensional effects.
BackColor	OLE_COLOR	Sets the Background color of the control.
BeepOnError	BOOL	Beep on validation errors. (Default - False)
BorderStyle	Integer	0 - The control has no visible border line. 1 - The control has a single-line border (default).
ContextMenu	Boolean	Enable or Disable the default menu associated with the control. Is dependent on the control type. (Default - True)
EnableCalendar	Boolean	Enable/Disable the Calendar control button.
Enabled	Boolean	Enable/Disable the control
Font	IFontDispatch	Sets the control's font.
ForeColor	OLE_COLOR	Sets the foreground color of the control.

Format	String	The format string that specifies the formatted output of date or time. By default this is set to the ViewLocale's (ViewLocale) long date format. For Time control it is set to the ViewLocale's default.
MilleniumMode	BOOL	Sets/Gets the MilleniumMode for input entry, which determines whether the year is 2 or 4 digits.
ResourceLocale	VARIANT	The locale to use for displaying errors and menu resources. This value could be one of the following: · A locale identifier. · A concatenation of language and country abbreviation.
ShowToolTip Style	Boolean Integer	Enable or Disable the Tool Tips for the control. (Default - True) 0 - Date/Time Control (Default) 1 - Date Control 2 - Time Control
Text	String	Returns/Sets text contained in the control object.
Value	OLE DATE	The actual date/time value represented by an 8 byte OLE Date.
ViewLocale	VARIANT	The ViewLocale that the control represents. This value could be one of the following: · A locale identifier. · A concatenation of language and country abbreviation. · Another locale object.
ViewLocaleMenu	Special	· Consists of the locale ids that are displayed when the context menu button is pressed, assuming the ContextMenu property is TRUE.

Table 5: Time Specific Properties

Name	Type	Description
Hour	Integer	The Hour value represented by the Date/Time
Minute	Integer	The minute value represented by the Date/Time
Second	Integer	The second value represented by the Date/Time

Table 6: Date Specific Properties

Name	Type	Description
Day	Integer	The Day value represented by the Date/Time
Month	Integer	The Month value represented by the Date/Time
Year	Integer	The Year value represented by the Date/Time

Methods

HRESULT GetDayOfWeek(short *pDayOfWeek, BSTR *pDayOfWeekStr)

Returns the day of week as an integer and a string representation.

HRESULT GetDayOfYear(short *pDayOfYear)

Returns the day of year as an integer.

Events

ViewLocaleChange

This event is generated when a view locale change has been requested by the user via the context sensitive menu.

DateTimeChange

Param: DATE NewDate

This event is generated when the user has modified the date via the user interface.

D) Locale Multilingual Component

The locale multilingual component is a non-visual control used to define the language/country/font and other information pertaining to a particular locale ID. On WindowsNT platform the developer can set the locale for the whole system (Default Locale) or for a particular thread, via the SetThreadLocale() API call. However, this level of locale specification is insufficient to allow for individualized localization of particular multilingual components. Thus, in accordance with the present invention, the locale multilingual component provides a finer granularity & encapsulates locale management on per locale basis, giving the capability for each class to have its own locale, via containment. Locale information can either reflect the information, as defined by the system, or it can reflect any

user overridden properties. This feature is controlled via a UseUserOverride (or similar) property. It also features enumeration of supported and installed locales on the current system. All other types of multilingual components are dependent on at least one locale multilingual component to provide the information about the requirements of a particular locale.

During initialization the locale multilingual component's default LocaleID is set to the default user supplied locale ID, such as provided by the operating system. The locale multilingual component then obtains from the operating system the appropriate formatting information for the locale, and makes this information available to other multilingual components. If this fails for some reason the system default locale ID is used. By default the locale multilingual component allows access to all the read only properties, which cannot be overridden.

Table 7: Class Name: MC.Locale Class Properties

Name	Type	Description
AnsiCodePage	Long	ANSI code page for the Locale. (R/O-read only)
CharSet	Short	Character Set representing the Code Page. (R/O)
CountryCode	Long	Country code, based on International phone codes, associated with the Locale. (R/O)
CountryName	String	Country name associated with the Locale. (R/O)
CountryNameAbbrev	String	Country Name abbreviation as specified in the ISO standard 3166. (R/O)
CurDecimalSeparator	String	Currency decimal separator string for the Locale.
CurFractionalDigits	Integer	Currency fractional digits for the Locale.
CurGroupingSize	Integer	The size for each group of monetary digits to the left of the decimal separator.
CurIntlSymbol	String	The international symbol characters specified by the ISO 4217.
CurNegativeStyle	Integer	One of the 16 negative currency modes to use to display negative monetary values. This value must be between 0 - 15.
CurPositiveStyle	Integer	One of the 4 positive currency modes to use to display positive monetary values. This value must be between 0

		- 3.
CurSymbol	String	Currency symbol for the current Locale.
CurThousandSeparator	String	The currency thousand separator string for the locale
EngCountryName	String	Country Name in English. Characters in the string are restricted to characters mappable into the ASCII 127 character Subset. (R/O)
EngLanguageName	String	Language name in English. Based on the ISO standard 639 with characters mappable into the ASCII 127 char subset. (R/O)
LanguageAbbrev	String	Language name abbreviation. Created by taking a 2 letter language abbreviation from ISO standard 639, and adding a 3rd letter to indicate sublanguage. (R/O)
LanguageId	Short	Language ID associated with the Locale. (R/O)
LanguageName	String	Fully localized language name associated with the Locale. (R/O)
Locale	VARIANT	The locale that the control represents. This value could be one of the following: · A locale identifier. · A concatenation of language and country abbreviation. · Another locale object.
NativeCountryName	String	Country Name, in native language, associated with the Locale. (R/O)
NativeLanguageName	String	Language name, in the native language, associated with the Locale. (R/O)
NegativeSign	String	String representing the negative sign. (R/O)
NumDecimalSeparator	String	Numerical decimal separator string for the Locale.
NumFractionalDigits	Integer	Numerical fractional digits for the Locale.
NumGroupingSize	Integer	The size for each group of numerical digits to the left of the decimal separator.
NumLeadingZeros	Integer	Numerical leading zeros in decimal fields. Possible values: 0 - No Leading Zeros 1 - Leading Zeros

NumNegativeStyle	Integer	One of possible negative number modes supported by NLS API.
NumThousandSeparator	String	The numerical thousand separator string for the locale
PositiveSign	String	String representing the positive sign. (R/O)
UseUserOverride	BOOLEAN	TRUE, if Locale user overrides should be used for getting locale information. Default is TRUE, i.e. use user overridden locale information.

Methods

SAFEARRAY EnumerateFonts()

Returns a SafeArray of font dispatch pointers, for all fonts associated with the Locale property.

SAFEARRAY Enumerate(Boolean LocaleType)

LocaleType:

0 : Supported Locales

1 : Installed Locales

Enumerates all the Locales supported by the system, and returns them as a safe array of Dispatch IDs of Locale objects.

LPDISPATCH GetDefaultFont()

Chooses and returns a font object, from a list of available fonts, that can be used to display the charset represented by the Locale property.

E) Numeric Multilingual Component

The numeric multilingual component furnishes capabilities to input numeric values, in a locale sensitive manner.

(a) Component Behavior

The numeric multilingual component provides input validation by letting the user input only numbers, negative sign, and or the decimal separator string. The display is formatted

according to the ViewLocale property, when the component loses focus. Changing the ViewLocale property resets the underlying currency formatting properties.

The underlying value of the numeral is set under two conditions:

1. Directly by setting the Value property
2. Implicitly after the user has entered data in the control, and it has been successfully parsed.

(b) Component Min/Max Ranges

The Minimum and Maximum properties are used to limit the range of numeric values that will be accepted by the component. Normally the Minimum and Maximum properties are both set to 0, which implies no enforcement of range limits (except of course and limitations of the underlying double data type). The following are the preferred features of the use of Minimum and Maximum properties, in conjunction with a SpinButton (increment/decrement control) and value Stepping properties:

- If the Minimum and Maximum properties are equal (but not 0), then only the Minimum value can be input via the control.
- If Minimum value is less than the Maximum value, and the Stepping property and the SpinButton properties are set, then clicking the up arrow will increment values in the control by the Stepping amount, and clicking the down arrow of the spin will decrement values, enforcing the range limits.
- If Minimum value is greater than the Maximum value, and the Stepping property and the SpinButton properties are set, then clicking the up arrow will decrement values in the control by the Stepping amount, and clicking the down arrow of the spin will increment values, enforcing the range limits.
- If the Minimum or Maximum properties is changed and the underlying Value property is no more within the range, the Value property defaults to the Minimum value.

(c) Input Validation

The parsing of input and validation occurs when the control loses focus.

Table 8: Class Name: MC.Numeric Class Properties

Name	Type	Description
BeepOnError	BOOL	Beep on validation errors. (Default - False)
DecimalSeparator	String	The decimal separator string used to

		display the output.
FractionalDigits	Integer	The number of fractional digits to display for the locale.
GroupingSize	Integer	The size for each group of numeric digits to the left of the decimal separator.
LeadingZeros	Integer	Specifies the leading Zeros in decimal fields. Possible values: 0 - No Leading Zeros 1 - Leading Zeros
Maximum	double	The maximum value of the limit range. Is used in conjunction with the Minimum property. If both Minimum and Maximum property values are 0, no range is implied. (Default – 0)
Minimum	double	The minimum value of the limit range. Is used in conjunction with the Maximum property. If both Minimum and Maximum property values are 0, no range is implied. (Default – 0)
NegativeStyle	Integer	One negative Numeric modes to use to display negative values. This value must be between 0 - 4. 0 : (1.1) 1 : -1.1 2 : -1.1 3 : 1.1 4 : 1.1
ResourceLocale	VARIANT	The locale to use for displaying errors and menu resources. This value could be one of the following: · A locale identifier. · A concatenation of language and country abbreviation.
SpinButton	BOOL	Enable/Disable Spin Button. (Default – False)
Stepping	Float	Used in conjunction with the Spin property only. If Spin is enabled then clicking on the Spin control will increment or decrement the control value by the amount specified in this property. (Default - 1)
ThousandSeparator	String	The thousand separator string used to display the output
Value	double	The actual numeric value represented by the control. (Default - 0)
ViewLocale	VARIANT	The ViewLocale that the control represents. This value could be one of the

		following: · A locale identifier. · A concatenation of language and country abbreviation. · Another locale object.

Events

ViewLocaleChange

This event is generated when a view locale change has been requested by the user via the context sensitive menu.

F) Resource Multilingual component

The Resource multilingual component enables an application to retrieve native resources like strings, bitmaps and icons based on the Locale property. Rather than hard coding strings in an application, the resource multilingual component provides a locale independent way of retrieving strings based on a Resource ID. A particular Resource ID can represent a string, bitmap or icon in different locales.

The resource multilingual component constructs a DLL name by concatenating the Language, Version and Application properties. Any request to retrieve a resource is fulfilled by acquiring the resource from this DLL. The DLL is searched in the following order:

1. Directory the Application was loaded from;
2. Current directory;
3. System directory;
4. Window's directory;
5. Directories in the PATH variable.

Table 9: Class Name: MC.Resource Class Properties

Name	Type	Description
Application	String	A unique application name string.
Language	String	The language string that represents the resource. 3 Letter ISO standard, i.e. ENU.
Locale	VARIANT	The locale that the control represents. This value could be one of the following: · A locale identifier.

		· A concatenation of language and country abbreviation. · Another locale object.
Version	String	The version string for the DLL file.

Methods

BSTR GetString(Long ResourceID)

Returns a string representing a ResourceID for the locale specified by the Locale property.

SAFEARRAY GetBitmap(Long ResourceID)

Returns a single dimension safe array of bytes which contains a binary representation of a bitmap, based on the Locale property.

SAFEARRAY GetIcon(Long ResourceID)

Returns a single dimension safe array of bytes which contains a binary representation of an icon, based on the Locale property.

BSTR FormatString(Long ResourceID, SAFEARRAY saVarArguments)

Formats a string represented by ResourceID, replacing format specifiers with strings contained in the array saVarArguments.

Events: None

G) Edit Multilingual Component

The edit multilingual component is a visual control for the purpose of inputting free format textual information. Fig. 5b illustrated the property sheet for defining the properties of the edit multilingual component. The multilingual component language and font is determined by the ViewLocale property. Depending upon the ViewLocale the input may either be displayed left to right or right to left or with BIDI (bi-directional text entry) functionality. The edit multilingual component ensures proper editing/selection of characters whether the characters are half or full width.

An edit multilingual component can be single or multiline. If a multiline edit multilingual component cannot display all lines of data in its window, the scroll bar appears automatically, either on the right or the left edge of the window, depending upon the ViewLocale property.

Table 10: Class Name: MC.Edit Class Properties

Name	Type	Description
MaxLength	Long	Sets the maximum number of characters that can be entered
Modified	Boolean	Indicates if the text has been modified in the control
MultiLine	Boolean	Selects SingleLine or MultiLine mode. (Default - False)
PasswordChar	String	A one character string that is displayed as a password character. If this property is set, the user sees the password character for every character inputted in the control.
ReadOnly	Boolean	Determines if the Text in the control can be edited. (Default - True)
ResourceLocale	VARIANT	The locale to use for displaying errors and menu resources. This value could be one of the following: · A locale identifier. · A concatenation of language and country abbreviation.
ScrollBars	Short	· 0 – None · 1 – Horizontal · 2 – Vertical · 3 – Horizontal & Vertical
ViewLocale	VARIANT	The ViewLocale that the control represents. This value could be one of the following: · A locale identifier. · A concatenation of language and country abbreviation. · Another locale object.

Methods: None

Events

KeyPress

This event is generated when a user presses a key in the control area.

ViewLocaleChange

This event is generated when a view locale change has been requested by the user via the context sensitive menu.

Text Multilingual Component

The text multilingual component is a non-visual control which provides string manipulation services. A new instance of the text multilingual component acquires its initial locale ID from the current user default locale. The string that is represented by the text multilingual component is associated with the Locale property, which may be overridden by the user. All string mappings and conversions are based on the Locale property.

Table 11
Class Name: MC.Text Class Properties

Name	Type	Description
Text	Long	The underlying string
ResourceLocale	VARIANT	The locale to use for displaying errors and menu resources. This value could be one of the following: · A locale identifier. · A concatenation of language and country abbreviation.
ViewLocale	VARIANT	The locale that the control represents. This value could be one of the following: · A locale identifier. · A concatenation of language and country abbreviation. · Another locale object.

Methods

Integer GetLength ()

Returns the character length of the underlying string.

Void TurnUpper()

Converts the underlying string to upper case.

Void TurnLower()

Converts the underlying string to lower case.

Integer Compare(Text objTarget)

Compares the current object with the target object objTarget, and returns a 0 if they match, -1 if source is less than target, and 1 if the target is greater than source.

Integer SetAt(Integer strIndex, Integer chValue)

Replaces the character at strIndex with the character value chValue.

Integer GetAt(Integer strIndex)

Returns the value of character at position strIndex in the string.

Void Append(Text objTarget)

Appends the string object objTarget to the underlying string. Use the Text property to access the underlying string.

Integer Find(Integer FindChar)

Returns the index at which the character represented by FindChar is found.

Events: None

In summary, the present invention provides various multilingual components that encapsulate intelligent behavior and data for dynamically altering the language, formatting, and presentation of data in a software application. The components may be used by an applications developer to provide robust internationalized software that may be easily adapted, either during development or at runtime, to the specific cultural or language requirements for different geographic regions where the software it distributed or used. The present invention encompasses the disclosed multilingual components, and any other component that encapsulates the information necessary to reformat its data for different locales dynamically, including special purpose components, such as the calendar component. Other components that come within the scope of the present invention, may include, for example, postal address multilingual component that provides a multilingual address data entry and presentation environment, and automatically formats addresses according to a selected locale's addressing format parameters; a telephone number multilingual component provides a multilingual telephone number data entry and presentation environment, and that automatically reformats a telephone number; and a drop-down list multilingual component that provides a programmable multilingual drop-down list.

We claim:

1. A computer program product for executing on a computer system from a computer readable medium, and for providing support for multiple languages or locales, the program product comprising:

a multilingual component encapsulating data identifying a plurality of locales, each locale having a defined a language and formatting parameters for presenting data by the multilingual component, the multilingual component adapted to provide both a development time selection of a current locale for defining the language and formatting parameters of the component and a runtime selection of at least one of the plurality of locales that automatically presents the data with the language and the formatting parameters of the selected locale.

2. The computer program product of claim 1, further comprising:

a development application including:

a plurality of different types of multilingual components, each type providing type specific functionality for presenting a specific type of data;

a user interface mechanism that enables user-selected ones of the different types of multilingual components to be placed in a container of an application, the container having a specified locale, wherein each multilingual component placed with the container automatically selects the container's locale as the selected locale for presenting and formatting of its data.

3. The computer program product of claim 1, further comprising:

a client application including at least one container having a predefined locale, the container containing at least one multilingual component which automatically selects the container's predefined locale as the selected locale for presentation and formatting of its data.

4. The computer program product of claim 1, wherein the multilingual component includes a view locale property storing an identifier of the selected one of the plurality of locales for the language and formatting of stored data, and a resource locale property storing an identifier of a selected one of the plurality of locales for a language and formatting of

portions of the display of the multilingual component itself.

5. The computer program product of claim 1, further comprising:

a user accessible context menu listing at least some of the plurality of locales supported by the multilingual component, and that is adapted for the user to select a listed locale as the selected locale for the multilingual component, whereby the multilingual component automatically reformats its presented data according to the language and formatting parameters of the selected locale.

6. The computer program product of claim 1, further comprising:

a plurality of locale multilingual components, each locale multilingual component storing a plurality of properties defining the language and formatting parameters for a specific locale, wherein a multilingual component obtains the language and formatting parameters for its selected locale from the locale multilingual component associated with the selected locale.

7. The computer program product of claim 1, wherein the multilingual component comprises:

a calendar multilingual component that displays a multi-year calendar including a month name and day of week names in the language and formatting parameters of the selected locale, and dynamically updates the language and formatting of the month name and day of week names in response to a change in the selected locale.

8. The computer program product of claim 1, wherein the multilingual component comprises:

a date/time multilingual component that displays a date or time string in the language and formatting parameters of the selected locale, and dynamically updates the language and formatting of the date or time string in response to a change in the selected locale.

9. The computer program product of claim 1, wherein the multilingual component comprises:

a currency multilingual component that displays a currency value including a currency symbol and an amount in the language and formatting parameters of the selected

locale, including a currency symbol and number separators, and dynamically updates the language and formatting of the currency value in response to a change in the selected locale.

10. The computer program product of claim 1, wherein the multilingual component comprises:

a numeric multilingual component that displays a number string in the language and formatting parameters of the selected locale, the formatting requirements including a thousands separator and a decimal point separator, and dynamically updates the language and formatting of the date or time string in response to a change in the selected locale.

11. The computer program product of claim 1, wherein the multilingual component comprises:

an edit multilingual component that displays user provided text string in the language and formatting parameters of the selected locale, and dynamically updates the language and formatting of the text string in response to a change in the selected locale.

12. The computer program product of claim 1, wherein the multilingual component comprises:

a resource multilingual component that displays application resources including at least one of error messages, menus, or dialogues, in the language and formatting parameters of the selected locale, and dynamically updates the language and formatting of the application resources in response to a change in the selected locale.

13. A method of providing an internationalized software product, comprising:

including in the software product at least one multilingual component encapsulating data identifying a plurality of locales, each locale having a defined a language and formatting parameters for presenting data by the multilingual component;

receiving during runtime a selection of at least one of the plurality of locales; and

automatically presenting the data with the language and the formatting parameters of

the selected locale.

14. The method of claim 13, wherein the data is a currency value, and automatically presenting the data with the language and the formatting parameters of the selected locale further comprises:

automatically determining a font and a currency symbol appropriate to the selected locale; and

automatically applying the determined font and currency symbol to the currency value.

15. The method of claim 13, wherein the data is a month calendar display including a month name and day of week names, and automatically presenting the data with the language and the formatting parameters of the selected locale further comprises:

automatically determining a month name and day of week names appropriate to the selected locale; and

automatically applying the determined month name and day of week names to the calendar display.

16. The method of claim 13, wherein the data is a date/time string, and automatically presenting the data with the language and the formatting parameters of the selected locale further comprises:

automatically determining a date and time format appropriate to the selected locale, the determined date format specifying an order of day, month and year date elements, and the determined time format specifying an order of hour, minute, and second elements; and

automatically applying the determined date and time format to the date/time string.

17. The method of claim 13, wherein the data is a number, and automatically presenting the data with the language and the formatting parameters of the selected locale further comprises:

automatically determining a number format appropriate to the selected locale, the determined number format specifying a thousands separator and a decimal separator; and

automatically applying the determined number format to the number.

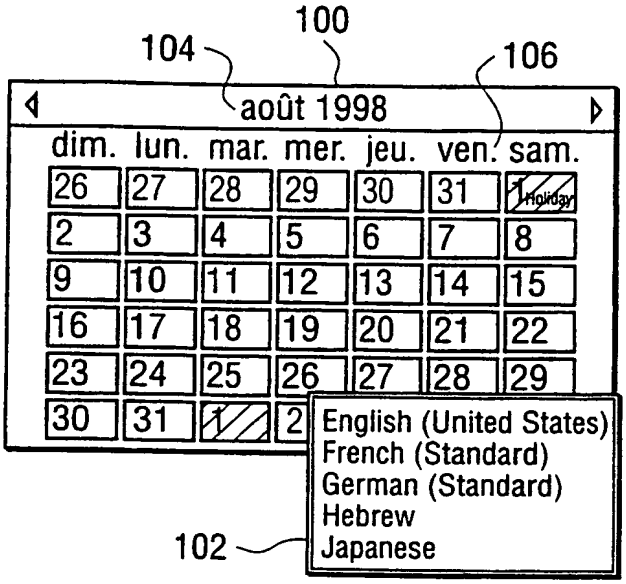


FIG. 1a

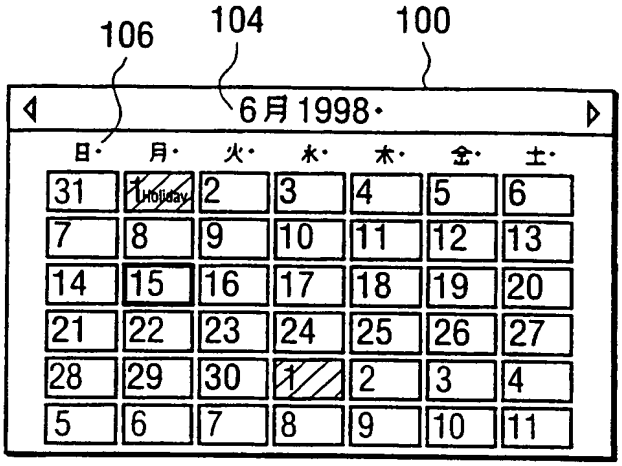


FIG. 1b

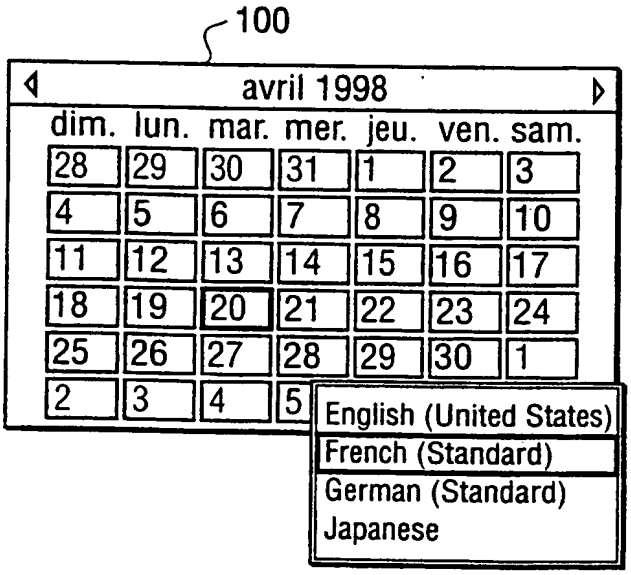


FIG. 1c

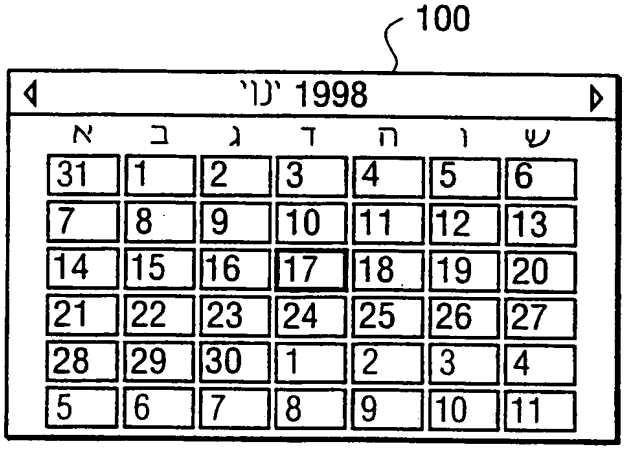


FIG. 1d

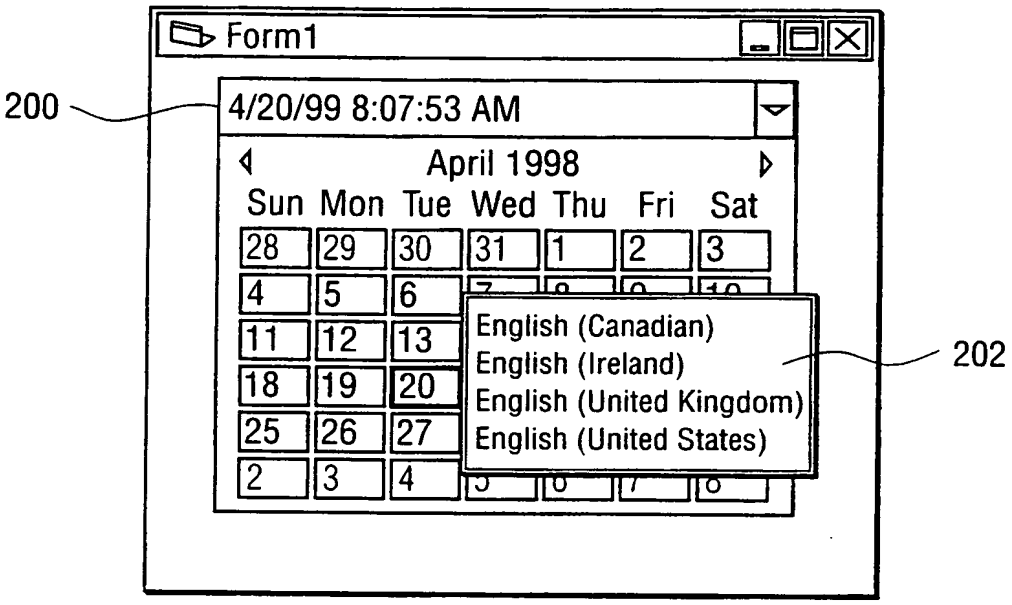


FIG. 2a

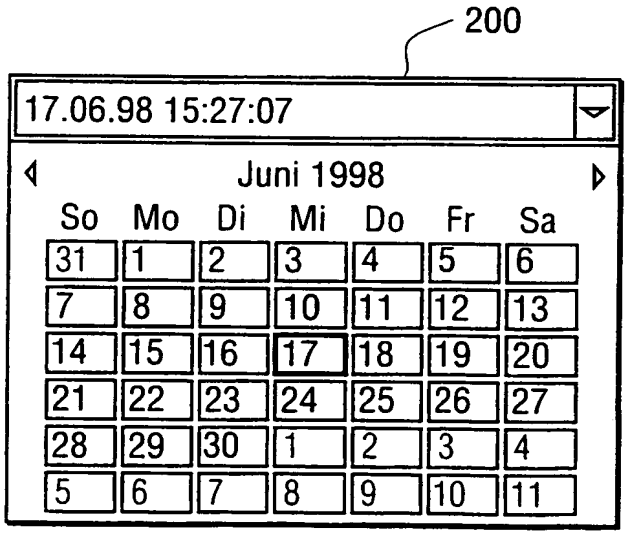


FIG. 2b

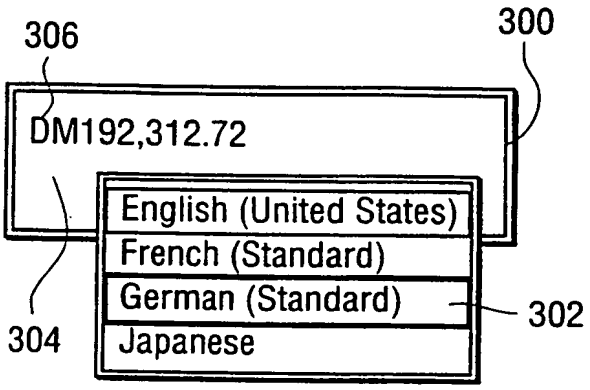


FIG. 3a

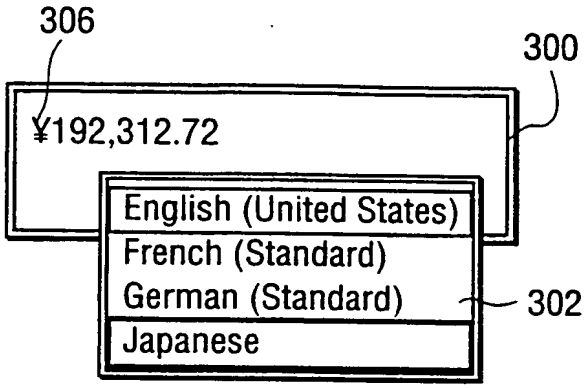


FIG. 3b

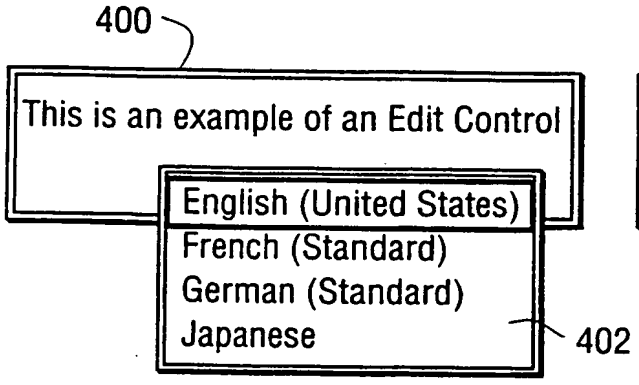


FIG. 4a

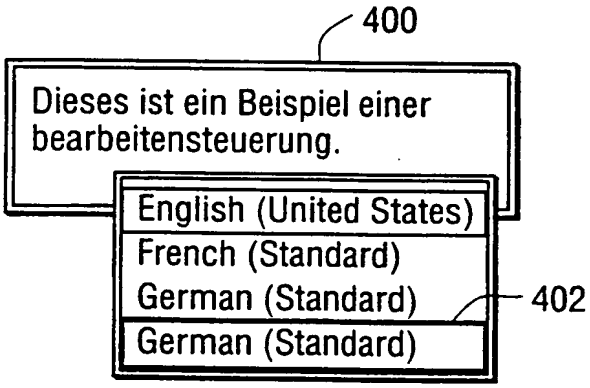


FIG. 4b

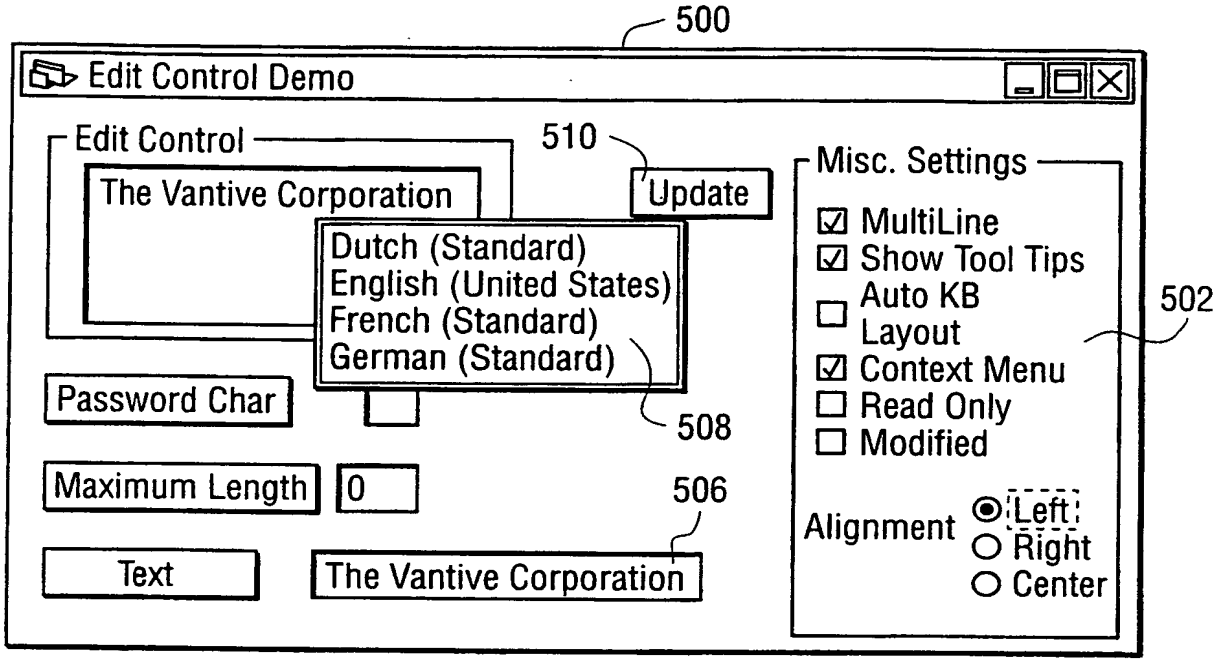


FIG. 5a

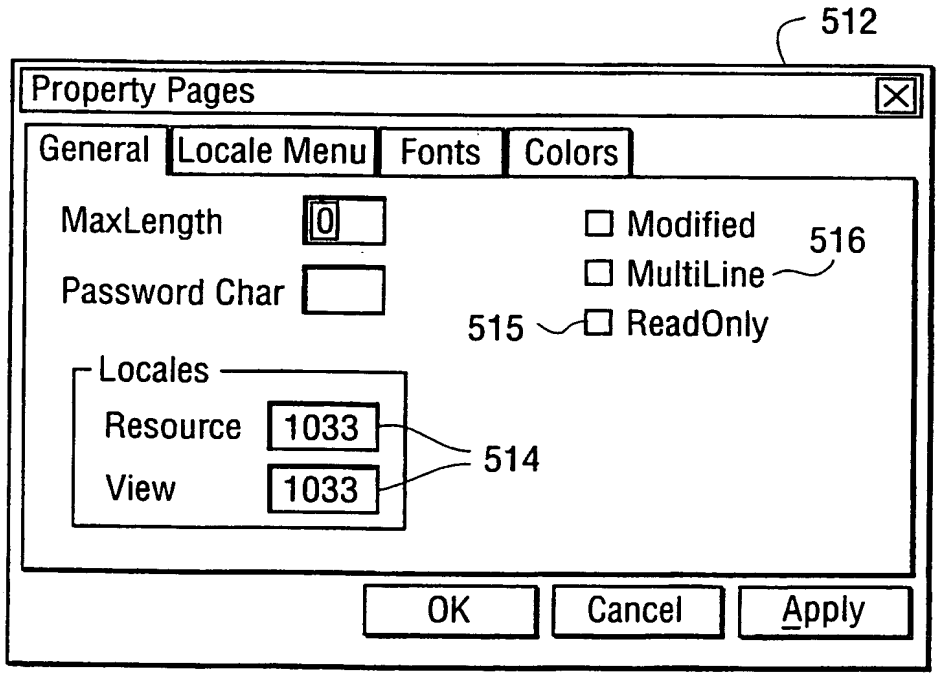


FIG. 5b

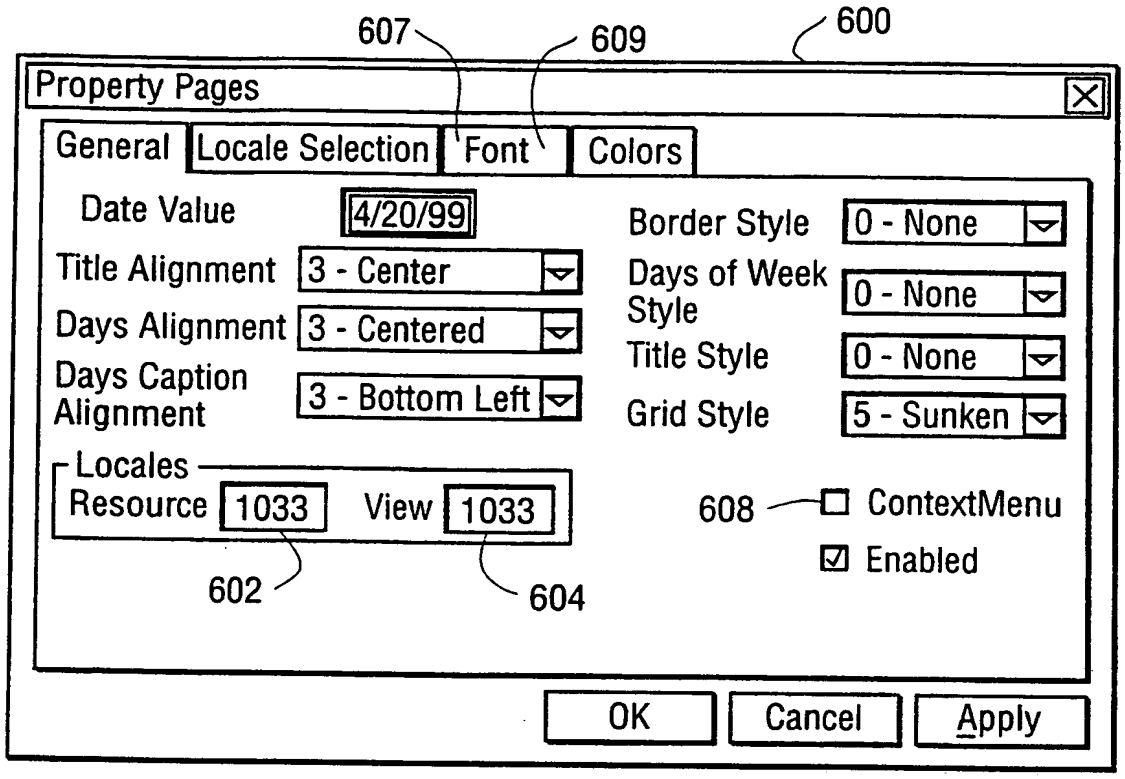

607
609
600
Property Pages
General
Locale Selection
Font
Colors
Date Value
4/20/99
Title Alignment
3 - Center
Days Alignment
3 - Centered
Days Caption Alignment
3 - Bottom Left
Locales
Resource
1033
View
1033
602
604
Border Style
0 - None
Days of Week Style
0 - None
Title Style
0 - None
Grid Style
5 - Sunken
608
ContextMenu
Enabled
OK
Cancel
Apply

FIG. 6a

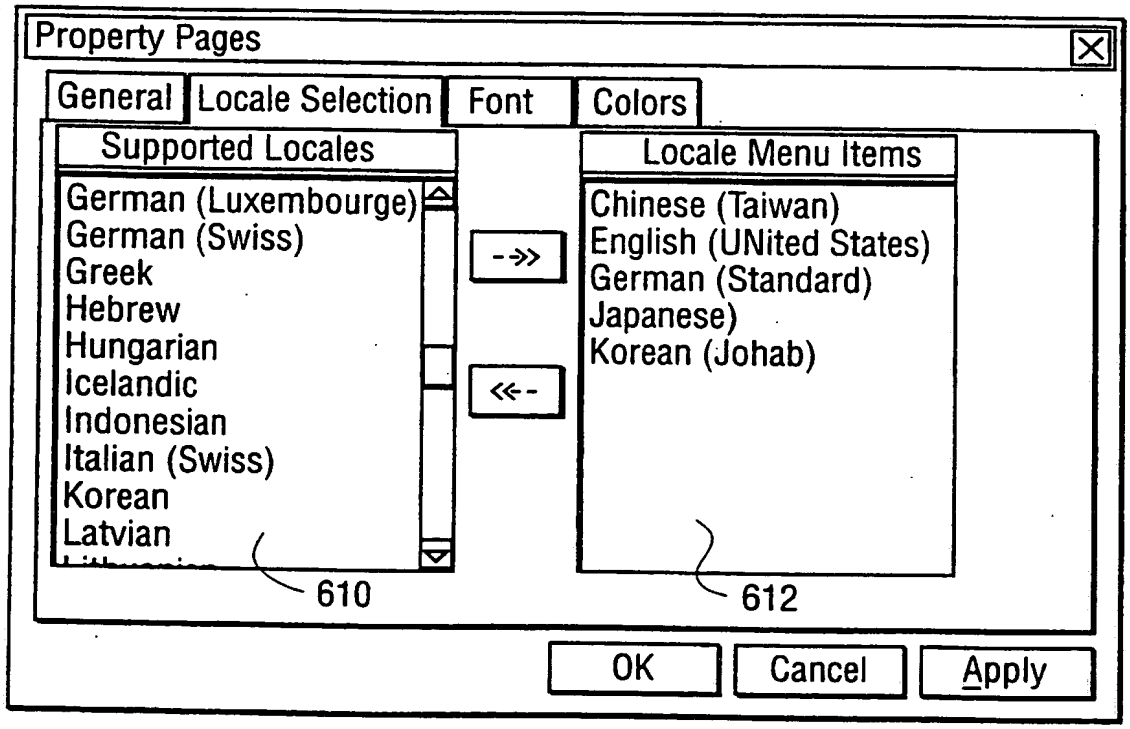

Property Pages
General
Locale Selection
Font
Colors
Supported Locales
German (Luxembourge)
German (Swiss)
Greek
Hebrew
Hungarian
Icelandic
Indonesian
Italian (Swiss)
Korean
Latvian
->>
<<-
Locale Menu Items
Chinese (Taiwan)
English (UNited States)
German (Standard)
Japanese)
Korean (Johab)
610
612
OK
Cancel
Apply

FIG. 6b

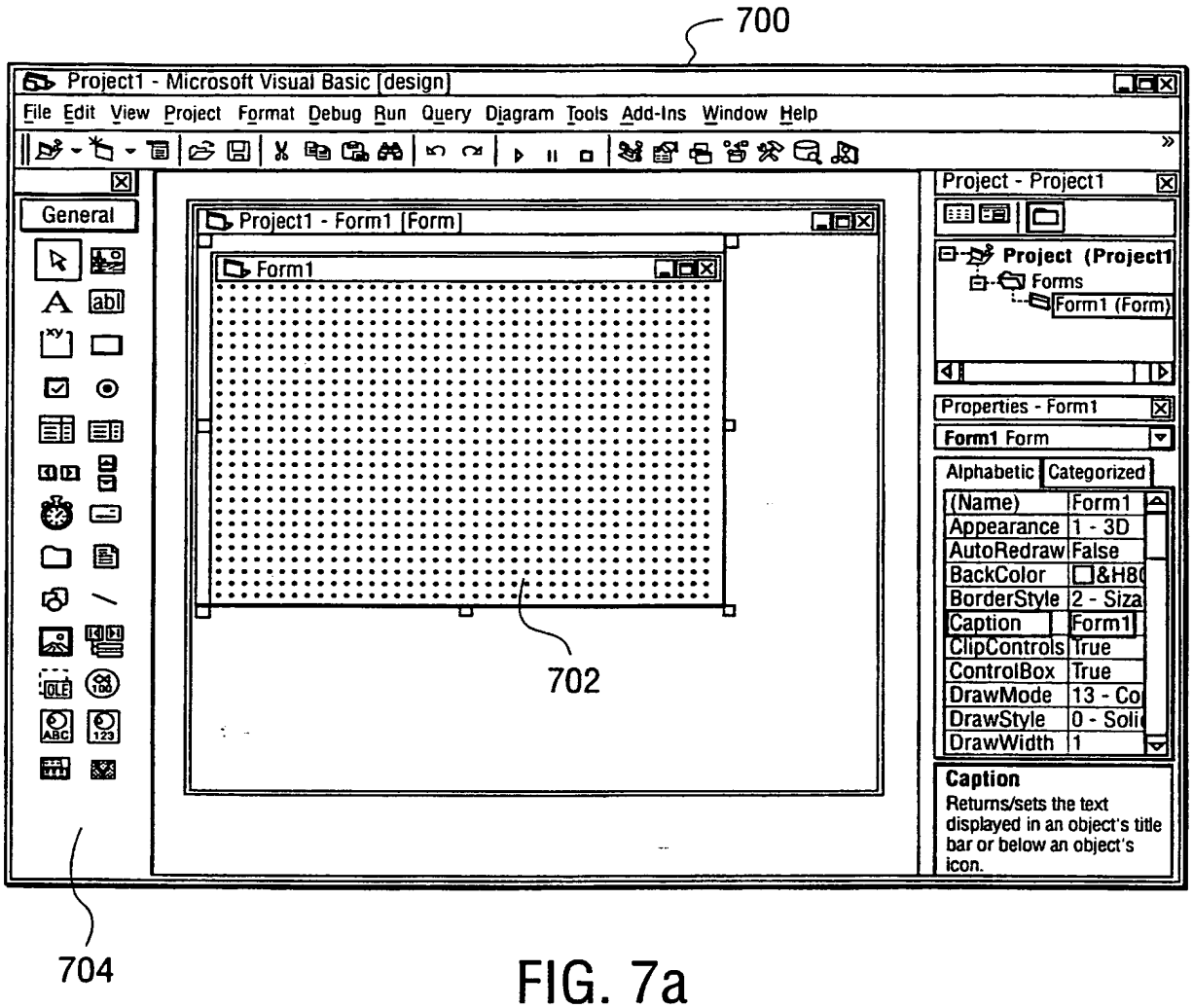

700
Project1 - Microsoft Visual Basic [design]
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help
General
Project1 - Form1 [Form]
Form1
702
704
Project - Project1
Project (Project1
Forms
Form1 (Form)
Properties - Form1
Form1 Form
Alphabetic
Categorized
(Name) Form1
Appearance 1 - 3D
AutoRedraw False
BackColor &H8
BorderStyle 2 - Siza
Caption Form1
ClipControls True
ControlBox True
DrawMode 13 - Co
DrawStyle 0 - Soli
DrawWidth 1
Caption
Returns/sets the text displayed in an object's title bar or below an object's icon.

FIG. 7a

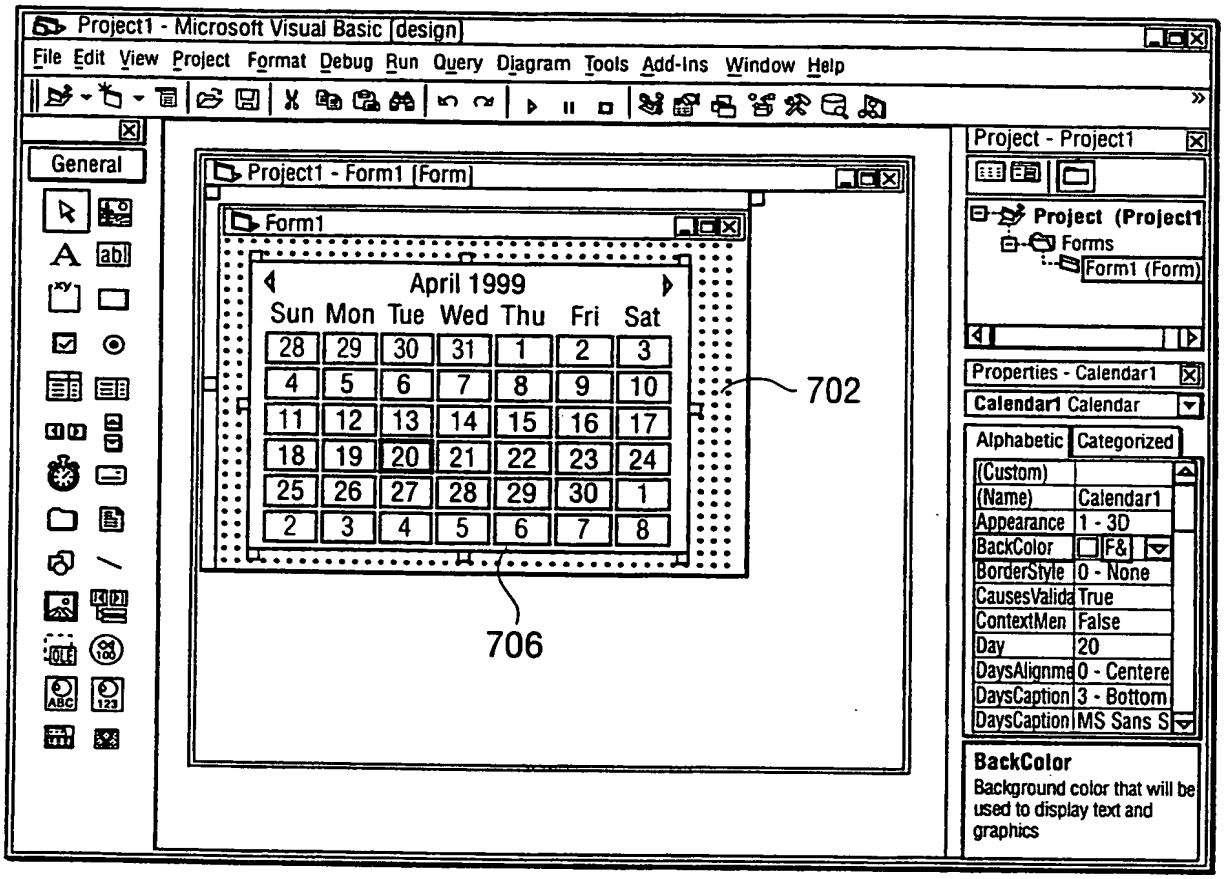

Project1 - Microsoft Visual Basic [design]
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help
General
Project1 - Form1 [Form]
Form1
April 1999
Sun Mon Tue Wed Thu Fri Sat
28 29 30 31 1 2 3
4 5 6 7 8 9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 1
2 3 4 5 6 7 8
702
706
Project - Project1
Project (Project1
Forms
Form1 (Form)
Properties - Calendar1
Calendar1 Calendar
Alphabetic
Categorized
(Custom)
(Name) Calendar1
Appearance 1 - 3D
BackColor F&
BorderStyle 0 - None
CausesValida True
ContextMen False
Day 20
DaysAlignme 0 - Centere
DaysCaption 3 - Bottom
DaysCaption MS Sans S
BackColor
Background color that will be used to display text and graphics

FIG. 7b

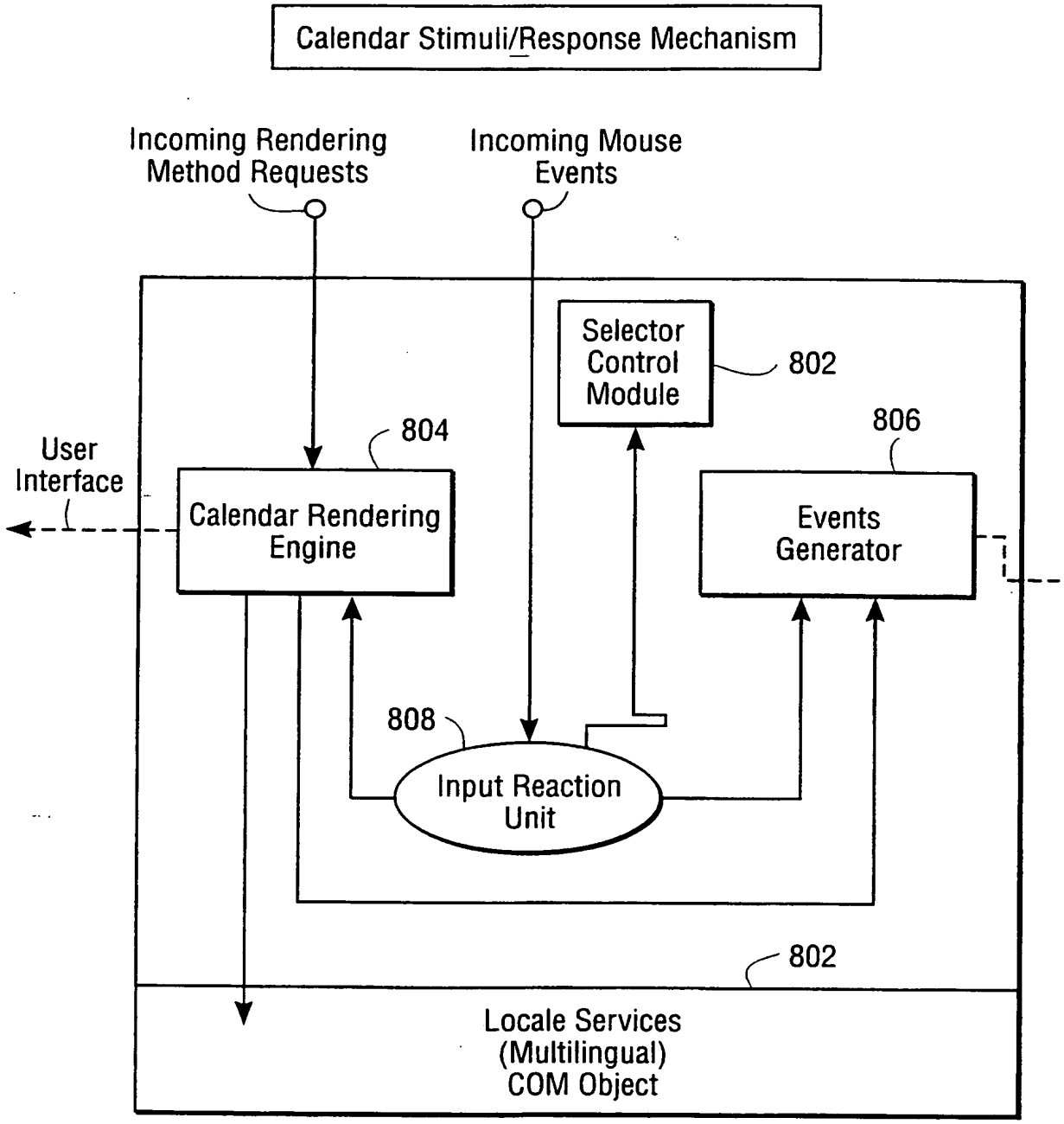


FIG. 8

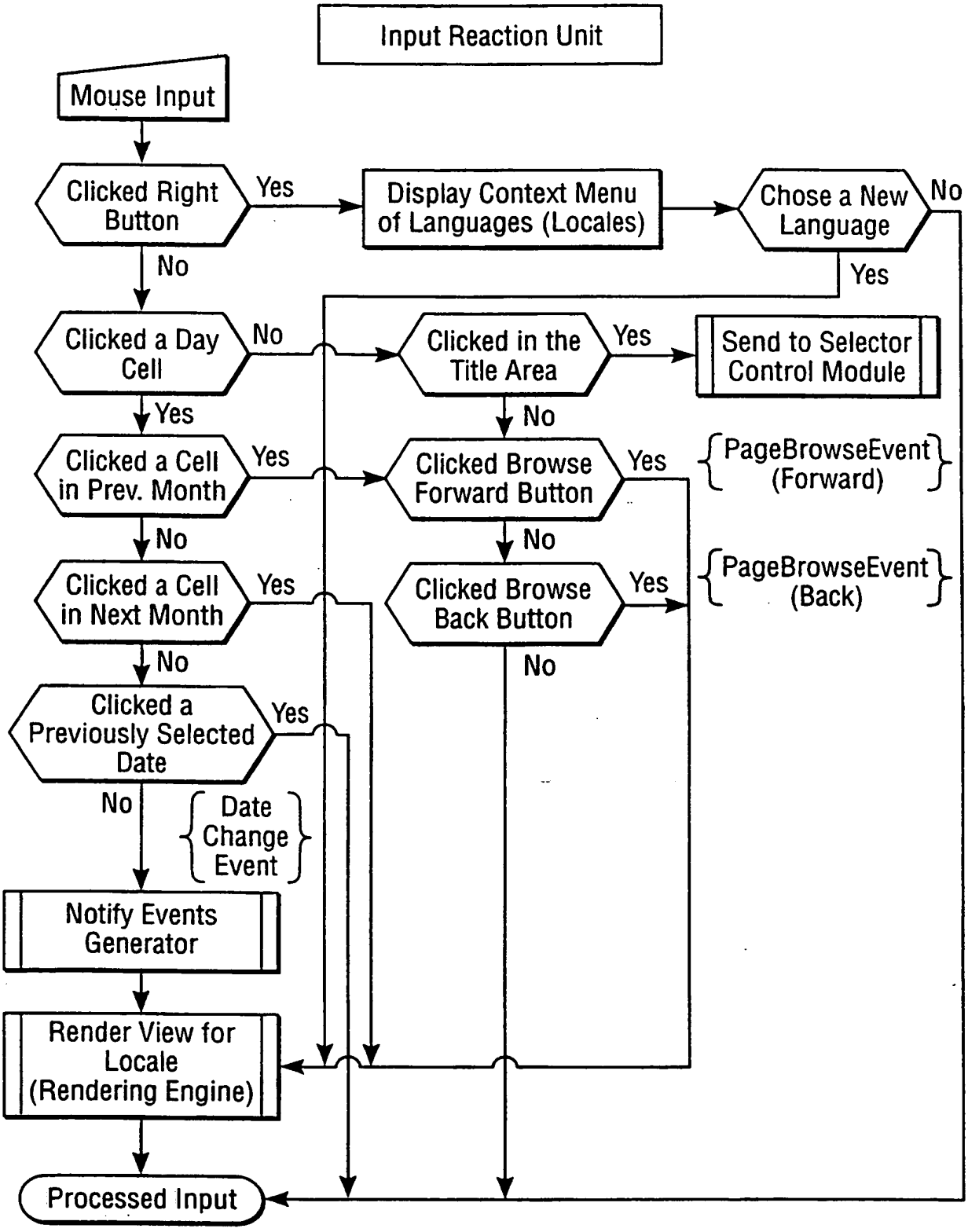

Input Reaction Unit
Mouse Input
Clicked Right Button
Yes
Display Context Menu of Languages (Locales)
Chose a New Language
No
Yes
No
Clicked a Day Cell
No
Clicked in the Title Area
Yes
Send to Selector Control Module
Yes
No
Clicked a Cell in Prev. Month
Yes
Clicked Browse Forward Button
Yes
PageBrowseEvent (Forward)
No
No
Clicked a Cell in Next Month
Yes
Clicked Browse Back Button
Yes
PageBrowseEvent (Back)
No
No
Clicked a Previously Selected Date
Yes
No
Date Change Event
Notify Events Generator
Render View for Locale (Rendering Engine)
Processed Input

FIG. 9

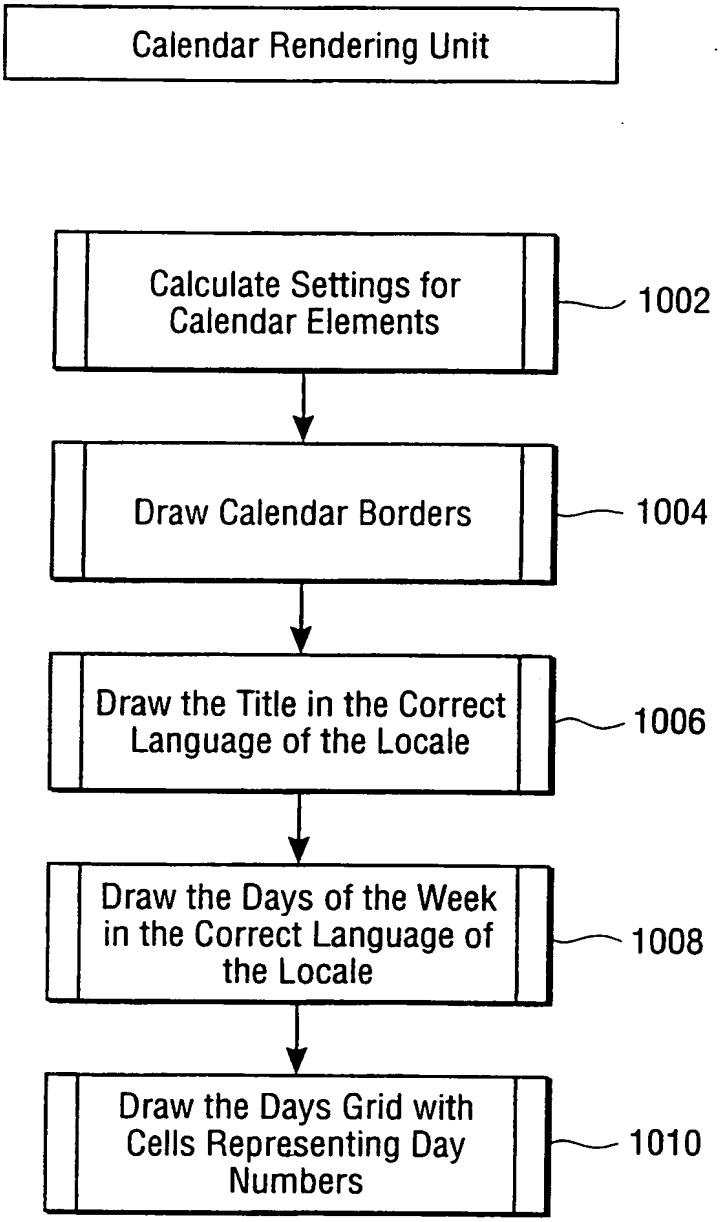


FIG. 10

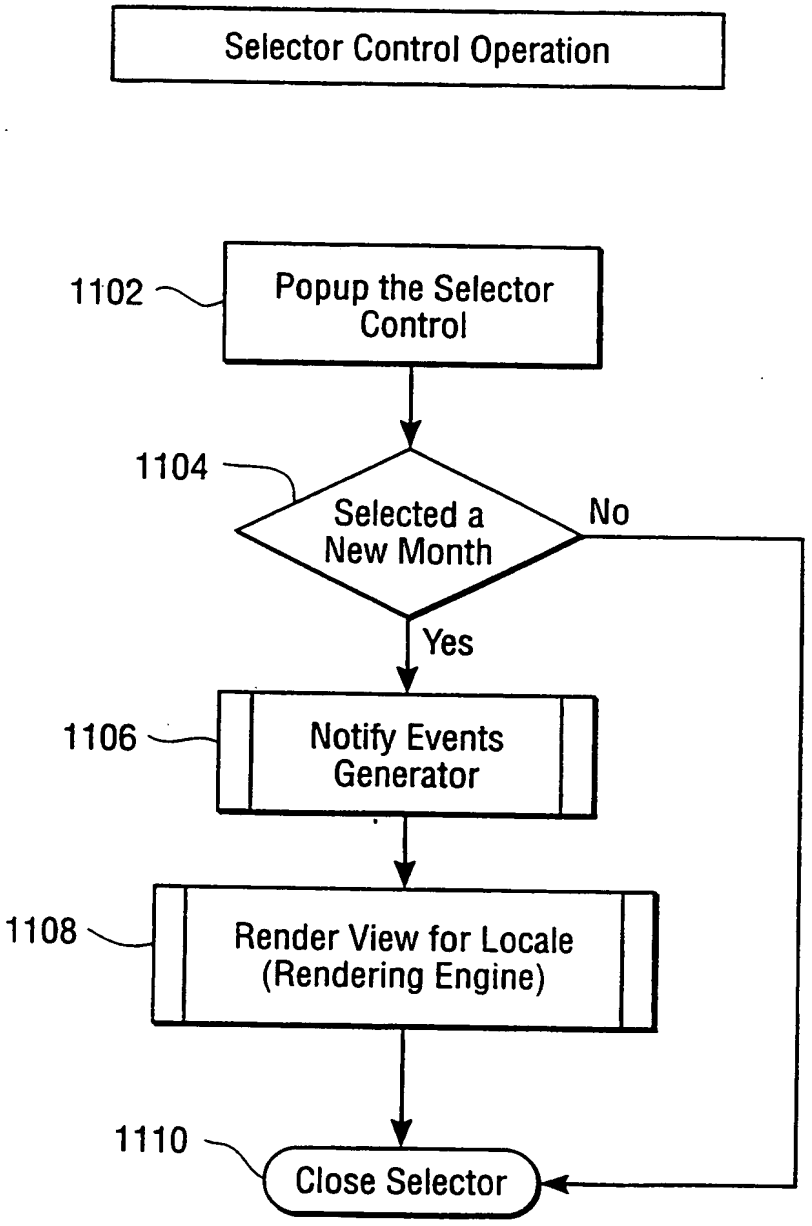

Selector Control Operation
1102
Popup the Selector Control
1104
Selected a New Month
No
Yes
1106
Notify Events Generator
1108
Render View for Locale (Rendering Engine)
1110
Close Selector

FIG. 11